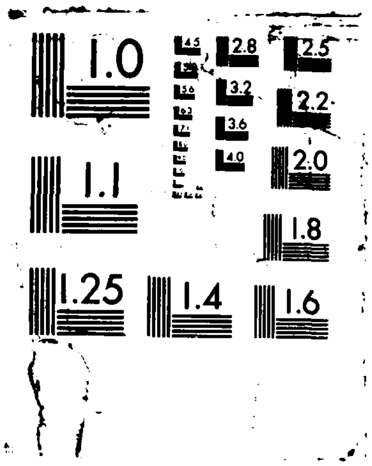
AD-A189 502 DEVELOPMENT OF A GENERAL PURPOSE COMMISSARY STORE 1/2
SIMULATION MODEL(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
UNCLASSIFIED R D MOULDER DEC 87 AFIT/GOR/ENS/87D-11 F/G 15/5 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5

(1)

AFIT/GOR/ENS/87D-11

DEVELOPMENT OF A GENERAL PURPOSE

COMMISSARY STORE SIMULATION MODEL

THESIS

Roger D. Moulder
Captain, USAF

AFIT/GOR/ENS/87D-11

Approved for public release; distribution unlimited

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A189 502

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED

1b. RESTRICTIVE MARKINGS:

2a. SECURITY CLASSIFICATION AUTHORITY:

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE:

3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GOR/ENS/87D-11

5. MONITORING ORGANIZATION REPORT NUMBER(S):

6a. NAME OF PERFORMING ORGANIZATION: School of Engineering

6b. OFFICE SYMBOL (If applicable): AFIT/ENS

7a. NAME OF MONITORING ORGANIZATION:

6c. ADDRESS (City, State, and ZIP Code):
Air Force Institute of Technology
Wright-Patterson AFB OH 45433-6583

7b. ADDRESS (City, State, and ZIP Code):

8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Air Force Commissary Service

8b. OFFICE SYMBOL (If applicable):

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER:

8c. ADDRESS (City, State, and ZIP Code): Kelly AFB TX 78241-5000

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification): See Box 19

12. PERSONAL AUTHOR(S): Roger D. Moulder, B.S., Capt, USAF

13a. TYPE OF REPORT: MS Thesis

13b. TIME COVERED: FROM ________ TO ______

14. DATE OF REPORT (Year, Month, Day): 1987 December

15. PAGE COUNT: 147

16. SUPPLEMENTARY NOTATION:

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
12	08	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number):
Computer Modeling, Queueing Theory, Simulation
Computer Simulation

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: DEVELOPMENT OF A GENERAL PURPOSE COMMISSARY STORE SIMULATION MODEL

Thesis Advisor: Joseph R. Litko, Major, USAF
Assistant Professor of Operations Research

Approved for public release: IAW AFR 190-1/.
Lynn E. Wolaver 24 Feb 88
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL: Joseph R. Litko, Major, USAF

22b. TELEPHONE (Include Area Code): (513)255-3362

22c. OFFICE SYMBOL: AFIT/ENS

DD Form 1473, JUN 86 — Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

UNCLASSIFIED

The purpose of this research was to develop a general purpose commissary store simulation model. This model would provide Air Force Commissary Service (AFCOMS) with an analytical tool to aid in managing Air Force Commissaries. Two of the issues AFCOMS is most interested in are minimizing customer waiting time in checkout lines and scheduling cashiers optimally. In order for the model to provide AFCOMS with the flexibility needed to investigate these performance measures, the following objectives had to be met: (1) Develop the model so that it is general enough to be used for any Air Force Commissary. (2) Make the model easy to use. (3) Develop a user's manual for potential users of the model. (4) Verify and validate the model. (5) Develop guidelines for data collection and analysis. (6) Develop statistical models which predict customer shopping times and checkout times. (7) Propose appropriate distributions for modeling customer arrival rates and the number of items customers buy. Keywords:

After the model was developed, data was collected at the Wright-Patterson AFB Commissary. During data collection and analysis, guidelines were developed so that potential users would be able to collect appropriate data to use with the model. The model was verified and validated and deemed to be a reasonable representation of a commissary store.

Computer modeling, Queueing theory, Computer simulation, Theses.

UNCLASSIFIED

AFIT/GOR/ENS/87D-11

DEVELOPMENT OF A GENERAL PURPOSE COMMISSARY STORE SIMULATION MODEL

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Roger D. Moulder, B.S.

Captain, USAF

December 1987

Approved for public release; distribution unlimited

Preface

The purpose of this research was to develop a general purpose commissary store simulation model. This model is needed by Air Force Commissary Service (AFCOMS) to help in managing the 139 commissaries at Air Force bases around the world. Two of the performance measures AFCOMS is most interested in investigating with the model are customer waiting times in checkout lines and scheduling of cashiers.

The model was developed so that it is general enough to model the basic operations of any commissary. Ease of use was also built into the model. The model was verified and validated and should be a useful management tool for AFCOMS.

I wish to thank my faculty advisor, Major Joseph R. Litko, for his expert guidance and genuine interest in my efforts. Also, thanks to Lt. Col. Thomas F. Schuppe for his helpful comments and suggestions as my thesis reader. Most importantly, thanks to my family, Alicia and Jon, my children, and my lovely wife Patricia. These are the people who were my inspiration to keep working when times were difficult. Without them, I could not have completed this work.

Roger D. Moulder

Table of Contents

List of Figures

List of Tables

Abstract

The purpose of this research was to develop a general purpose commissary store simulation model. This model would provide the Air Force Commissary Service (AFCOMS) with an analytic tool to aid in managing Air Force commissaries. Two of the issues AFCOMS is most concerned with are minimizing customer waiting time in checkout lines and scheduling cashiers optimally. In order for the model to provide AFCOMS with the flexibility needed to investigate different management strategies involving these two performance measures, the following objectives had to be met :
(1) Develop the model so that it is general enough to be used for any Air Force Commissary. (2) Make the model easy to use. (3) Develop a user's manual for potential users of the model. (4) Verify and validate the model with data collected at the Wright-Patterson AFB Commissary. (5) Develop guidelines for data collection and analysis. (6) Develop statistical models which predict customer shopping and checkout times. (7) Propose appropriate distributions for modeling customer arrival rates and the number of items customers buy.

After the model was developed, data was collected at the Wright-Patterson AFB Commissary. During data collection and analysis, guidelines were developed so that potential users would be able to collect appropriate data to use with the model.

The model was verified and validated with the data collected at the Wright-Patterson AFB store. The model was deemed to be a reasonable representation of a commissary store.

Finally, recommendations for further studies and model improvements were made which include further statistical studies of shopping and checkout times and automating the calculation of the number of runs of the model that must be made in order to reach a specified accuracy.

DEVELOPMENT OF A GENERAL PURPOSE COMMISSARY STORE SIMULATION MODEL

I. Introduction

Background

The Air Force Commissary Service (AFCOMS) is tasked with providing excellent commissary service to Air Force members and other authorized shoppers. This task is worldwide since Air Force bases are located throughout the free world. This is an enormous undertaking which involves the management of 139 stores and annual sales of $2.2 billion (26:115-116).

Unlike retail grocery stores, commissaries are not in business to make a profit. Commissaries provide a service, or a benefit, to the military community. Recent surveys of commissary patrons show that military members rank the commissary benefit as a main reason for choosing the Air Force as a career (26:115). AFCOMS recognizes the importance Air Force members place on the commissary benefit, and management at all levels in the Air Force Commissary Service continually strives to improve the quality of service in all Air Force commissaries.

In addition to providing an attractive benefit to military members, commissaries also help the Air Force save money. If commissaries were not operated, military pay would have to be increased so that military members could subsist

on the local economy. Commissary shoppers usually spend about 20% less for groceries at commissaries than they would have to spend for the same items at local groceries. Given that the total amount of goods sold at commissaries per year is approximately $2 billion, commissary patrons save about $400 million a year. This implies that if commissaries were closed, the Air Force would need to increase the pay of its members substantially if they were to maintain their current standard of living. The total cost of the pay increase would need to be $400 million, compared to the $200 million that it costs the Air Force to operate all of its commissaries. Therefore, by operating its own stores, the Air Force is providing the same benefit to its members at half the cost of merely increasing pay (8).

Since providing excellent commissary service to authorized shoppers, instead of making a profit, is the goal of the Air Force Commissary Service, commissary store managers face a variety of operational problems which are not common in the retail grocery business. Consequently, there are no industry guidelines to solve some of these problems. For example, a commissary faces extremely heavy shopping traffic around military pay days. The commissary store manager wants to ensure that all customers are able to shop and check out in the shortest time possible. Conversely, a retail grocery manager wants to keep customers in the store as long as possible in order to maximize the amount of groceries bought.

Commissary Service management feels that one way it can provide the best commissary service possible is to minimize the amount of time customers must wait in check out lines. To achieve this goal, commissary managers must be able to have the right number of personnel on duty to handle the workload. Given that each store is allocated a fixed budget for the hiring of personnel such as cashiers and stockers, managers have to schedule these workers in a way so as to maximize worker utilization and meet the goal of minimum customer waiting time simultaneously. Management must also consider other factors which influence a commissary's ability to provide the quality of service customers expect. Two of these factors are queueing line configurations and customer arrival rates.

Specific Problem

The unique goals and problems of the Air Force Commissary Service place unusual demands on all levels of management in the Air Force Commissary Service. Presently, there is no scientific or analytical tool available which management can use as an aid in making the decisions that will solve the problems faced in managing Air Force commissaries (8).

Research Objective

The purpose of this research was to develop a general purpose store simulation model which Air Force Commissary Service can use as an analytical tool to aid management in

making policy and operational decisions at Air Force commissaries worldwide.

Subsidiary Objectives

In order to reach the above research objective, the following subsidiary objectives had to be met.

1. Develop the model so that it is general enough to be used for any Air Force Commissary.
2. Make the model user-friendly.
3. Develop a user's guide for potential users of the model.
4. Validate the model with data collected at the Wright-Patterson AFB Commissary.
5. Develop guidelines for data collection at any Air Force Commissary so that data can be collected for use with the model.
6. Develop statistical models which predict customer shopping times and checkout times.
7. Propose appropriate distributions to use in modeling customer arrival rates and the number of items a customer buys.

Scope and Limitations

In order to validate the model, data was collected at the Wright-Patterson AFB Commissary. During the data collection at Wright-Patterson AFB, guidelines were developed to help future users collect the appropriate data to use the model.

Plan of the Report

Chapter I has provided an introduction to this research effort. The background, specific problem, research objective,

subsidiary objectives, and scope and limitations were discussed. Chapter II is the literature review and includes discussion of modeling, simulation, user-interaction with computers, and congestion modeling. Chapter III will discuss the development of the simulation model. Chapter IV provides the details of the data collection and analysis. Chapter V will cover model verification and validation and presents sample output of the model. To finalize the report, Chapter VI will discuss results, conclusions, and recommendations.

II. Literature Review

Overview

The main points of interest in the literature review were the following : 1) modeling in relation to retail marketing, grocery stores, and military commissaries; 2) simulation in relation to retail marketing, grocery stores, and commissaries; 3) user interaction with computers and user confidence in modeling; and 4) congestion modeling. Each point will be discussed in the order listed.

Modeling

Modeling enjoys widespread use in business and science. Models are used to investigate the behavior of physical and procedural systems. Physical systems are relatively easy to model since there are physical laws available that pertain to such systems. On the other hand, procedural systems are difficult to model because of the following reasons : 1) fundamental laws are not available; 2) procedures are difficult to describe; 3) policies are difficult to quantify; 4) randomness is present; and 5) the human factor is present. Procedural models are the main interest of management science and operations research analysts (21:7). Since a commissary is a procedural system, modeling techniques for such systems were of primary concern in this part of the literature review. Both theoretical and applied techniques were reviewed.

An example of how to apply the techniques of modeling was seen in a thesis that dealt with modeling an Air Force commissary (4). In this study the authors used simulation to investigate how various queueing configurations of the checkout lines in the Wright-Patterson AFB Commissary affected the average waiting time customers spend in the checkout lines. The authors concluded that single queue configurations consistently reduced customer waiting time in checkout lines (4:72). The conclusion of this study was that single line queueing reduces customer waiting time in checkout lines. Jones also supports this conclusion in his article by stating that over several years of study that single queues have been shown to reduce waiting time in lines (12:90).

In their thesis, Dorough and Holliway were interested in the time customers spent waiting in line to check out (4). It is easily seen that commissary customers are actually going through a series of queues when shopping at a commissary. For example, a customer may have to wait in line for any of the following reasons : 1) to obtain a parking spot; 2) to have their identification card checked; 3) to obtain a shopping cart; or 4) to checkout. All of these waiting lines are queues.

Since the operation of a commissary can be viewed as a tandem of queues, a more general model than the one developed in the above thesis was needed by AFCOMS to explore operational questions that deal not only with checkout

queues, but with all of the operating aspects of a commissary. The model that was developed had to be general and flexible. The choice of which type of model to develop had to be explored. Two choices were considered--analytic models and simulation models.

Modeling a queueing system with analytic techniques is possible when the probability distribution of the system is assumed to be steady-state. Steady-state means the system remains the same or is stationary over time (7:530). This is clearly not the case of a commissary since customers are arriving at varying rates throughout the day. This non-stationary characteristic of commissaries implies that analytic techniques are not the best choice for this research effort. Additionally, a commissary is a system with multiple queues, each with different arrival rates and service rates. This multi-stage property of commissary systems suggests that simulation is the best method with which to model a commissary since a simulation model describes the individual events of the individual components of a system rather than describing the overall behavior of a system directly (7:796).

Another reason simulation is the correct modeling technique for this research effort is that simulation is a way of experimenting with proposed systems without actually implementing them (7:826). AFCOMS management is interested in obtaining a tool that will provide flexibility in considering different configurations of commissaries. A

simulation model will provide AFCOMS with the ability to experiment with different configurations of new commissaries that will be built and of existing stores that are to be remodeled. Analytic techniques could be used to model overall behavior of a commissary but would not provide the flexibility AFCOMS needs.

HQ AFCOMS/XP is currently conducting a data collection effort to help in measuring the levels of labor needed to meet AFCOMS standards of customer service. The data being collected consists of hourly customer arrival rates, customer arrivals to queue lines, time customers spend in queue lines, and checkout times (11). This effort will be ongoing until 1988 at which time the data will be analyzed. Once the data is analyzed, the model developed during this thesis effort can be used to help AFCOMS management see how improvements can be made in such areas as evaluating manpower requirements, sizing stores' checkout capabilities, setting operating hours,and planning new stores.

Simulation

Simulation is the next major area that was researched in the literature. Pritsker defines computer simulation as follows:

> In its broadest sense, computer simulation is the process of designing a mathematical-logical model of a real system and experimenting with this model on a computer. Thus simulation encompasses a model building process as well as the design and implementation of an appropriate experiment involving that model [22:6].

Simulation is one way of modeling systems. If a system is extremely complex or stochastic in nature, it may be impossible to solve the system analytically. Simulation is a suitable tool to use when systems cannot be modeled analytically (6:381-382).

Since simulation is experimental, the analyst must plan the study by deciding the major parameters to be varied, the number of cases to be conducted, and the order in which to make the runs (5:53-55).

Due to its experimental nature, simulation has certain drawbacks (5:42). One drawback is that simulation gives specific solutions instead of general solutions. Several runs of a model will have to be done to understand the system under study. If an analyst chooses to use simulation in a study, plans must be made to treat the study as a series of experiments. Another drawback deals with optimization (6:381). Unfortunately, there is no guarantee that an optimal solution can be found even though the experimenter carefully designs the study. Many times the decision maker will ultimately choose the best of several alternatives offered by a simulation study, but the best alternative may not be the optimal solution. Simulation is nevertheless regarded as a valuable tool in systems analysis (5:42-43; 6:381-382).

Gordon says that simulation is an iterative process by which a model is refined. A model should start simple and become more complex at each iteration (5:53-55). Once

the basic system has been well defined by the model, more detail can be added to enhance the output of the simulation.

There is considerable disagreement over the use of simulation and the process of building simulation models. Standardization of the techniques of simulation is becoming a topic of intense study (25:66).

> When simulation first became feasible because of the advent of high speed computers, it was commonly believed that any model was better than no model at all. Likewise, it was believed that simulation was so simple that anyone could do it. It is a testimonial to the advancement of the profession that little credence is now given to such views. Even more, the areas of potential importance in simulation modeling have been enumerated and endeavors made to develop standards for them. It remains only to create a cadre of modelers capable and desirous of implementing those standards, rigorously and thoroughly [25:70].

Simulation is still being refined, but the results of simulation studies are still valid and useful.

Simulation has been used in other studies to investigate operational problems in commercial retail stores. In a study by Paul, a simulation model was used to show how to develop schedules for staffing service activities (19:206). A crucial aspect of this study was the involvement of supervisors and management in collecting data. The result was a model that was an effective scheduling tool (19:206-218).

Developing a simulation model involves establishing the model structure and supplying the data for the model (5:6-7). Establishing the model structure includes determining the system boundary and identifying the entities, the attributes

of the entities, and the activities which take place in the system (5:6). A good modeler draws the decision maker into the model formulation process for three reasons : 1) to ensure proper problem formulation; 2) to assist in determining details of the model; and 3) to set the stage for implementing results of the analysis (21:8). If the decision maker is involved in the development of the model, acceptance of the model as a valid tool is more likely.

As discussed earlier, analytic techniques could be used to model the overall system behavior of a commissary. However, it was decided to use simulation as opposed to analytic techniques in this research effort because : 1) commissaries are systems with individual components that require separate description; 2) the multiple queues in commissaries have time varying arrival and service rates; 3) the service rates are not exponential; and 4) simulation offers greater flexibility in exploring operational considerations such as scheduling workers or configuring checkout capabilities in stores.

User Interaction

The third main point of this literature review was user interaction with computer models. User confidence in simulation models must be established before decision makers will accept the output of the models. Model output must augment and complement the user's own knowledge of the system (21:14). Additionally, models are tools which must be

implemented in a way that will gain the confidence of the user (15:74).

The use of computers in solving operations research problems is now common. Unfortunately, there still exists skepticism about mixing the methodologies of management science and office automation. This skepticism is due in part to the fact that modeling in the past has been cumbersome and unfriendly to users.

> ...a basic issue in Management Science(implementation) is simply what decision support system will be within user expectations and needs, and yet will evolve with the user group to enlarge their felt needs for such support [27:74].

Involving the ultimate user of the model in the actual development of the model is one way the analyst is able to establish user confidence (21:8).

After reviewing the literature, it was clear to the author that user confidence in computer simulation models is established only if results obtained from the models are valid and if the models are easy to use. Ease of use was therefore a main consideration in the development of the simulation model during this research effort.

Congestion Modeling

The final area of the literature review dealt with modeling of congestion in stores or on highways. One proposed technique to model congestion in stores is to represent the amount of time a shopper spends in a store by a function in which shopping time depends on the number of persons in the

store (5:8). Similarly, traffic congestion has successfully been modeled by representing the time a car spends in traffic by a function in which driving time depends on the number of cars on the highway. (3:439). In other traffic congestion research, Ben-Akiva proposed to predict temporal distribution of volumes and delays at bottlenecks and to analyze impacts of alternative measures to alleviate peak period congestion. Simulation experiments showed several policies that could reduce peak period congestion with explicit information on the queues and delays in critical points (1:164).

Summary

This chapter has discussed the literature review for this research effort. First, modeling was described. Next, simulation was defined. The applicability of using analytic models as opposed to simulation models was also pointed out. The third main point--user-interaction--was discussed next. The importance of potential users having confidence in a model cannot be overemphasized. Finally, the idea of congestion modeling was shown to have been accomplished in studies conducted in the transportation field. Also, a technique to model congestion in stores was presented.

The next chapter discusses the development of the model to include the methodology employed and a description of the model.

III. Model Development

Overview

The overall objective of this research was to develop a simulation model that would be useful to management in Air Force Commissaries. A subsidiary objective was to develop the model so that it would be general enough to simulate the basic operations of any commissary. Another subsidiary objective was to make the model user-friendly. The user interface is therefore menu driven which makes the model easy to use. The methodology employed in developing the model and a description of the model are discussed below.

Methodology

The model was developed with ease of use as a main consideration. Analysts at HQ/AFCOMS are intended to be the primary users of the model. However, the model was developed so that once an analyst has collected data and made appropriate changes to the model as described in Chapter IV and Appendix B, managers at commissary stores can also use the model.

Early in the development stage, commissary officials and managers were interviewed so that the model would also be realistic (9; 12; 20). During these discussions with AFCOMS management, the topic of scheduling cashiers was always the main point. Since store managers have a limited budget with which to hire cashiers, it is extremely important to schedule these workers so that customers are served in a timely

manner. Timely in this context refers to the goal of Air Force Commissary Service to see that no customer has to wait in a checkout line for more than fifteen minutes.

Since the main interest of management is to schedule workers optimally, the model was developed to simulate the flow of customers through a commissary. A customer's flow through a commissary was modeled as follows :

1. obtain a parking spot;
2. pass through the identification card check point;
3. stamp a check;
4. obtain a shopping cart;
5. shop;
6. join a queue line to wait for checkout;
7. return to car and leave;

Of course, not all customers will write checks since some will pay cash. Also, some customers will be express line customers. The diagram in Figure 1 illustrates the flow of customers through a commissary.

The above phases of customer flow through a commissary were used as the basic framework of the model. The SLAM simulation language was used to model this process (22).

In order to make the model general enough to represent any Air Force commissary, FORTRAN subroutines were used in conjunction with SLAM. The flow of customers through a commissary as described above is consistent in all commissaries. The general configuration of commissaries is not the same everywhere. For example, some commissaries have

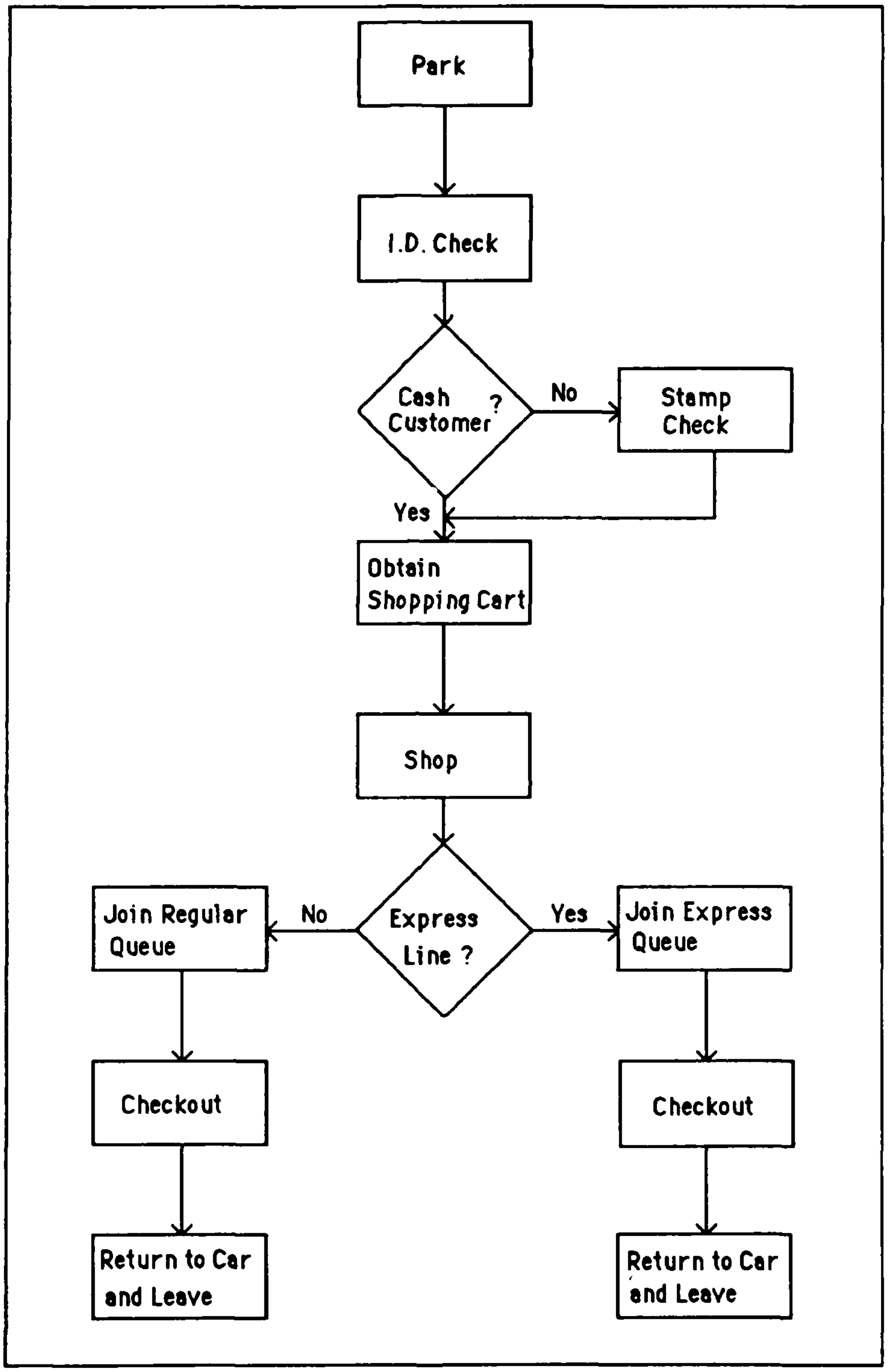


Figure 1. Customer Flow Through a Commissary

as many as 35 checkout stands while others have as few as one (20). Consequently, the number of shopping carts, parking spots, queue lines, number of hours open, and number of cashiers on duty will vary from store to store. The model was developed so that these parameters can be set by the user.

Commissaries also have operational differences. Some stores may have single queueing systems while others may have multiple queueing systems. Some examples of possible queueing systems are shown in Figures 2, 3, and 4. Figure 2 shows a queueing system with one waiting line and multiple servers. Figure 3 depicts a queueing system with multiple waiting lines and multiple servers. Note that the number of waiting lines is less than the number of servers. Like the system in Figure 3, the queueing system shown in Figure 4 has multiple waiting lines and multiple servers, but there is a one-to-one correspondence between each waiting line and each server. That is, each queue line has a dedicated server which services only that line.

An interesting observation can be made about the queueing systems depicted in Figures 2,3, and 4. If jockeying is allowed, that is, if customers are allowed to switch queue lines, then the queueing systems in figures 3 and 4 become equivalent to the system shown in Figure 2.

It should be noted that jockeying may not be possible in some stores. For example, if the queue lines are in the shopping aisles, then customers may not be able to see if another queue line is moving faster than the one they are

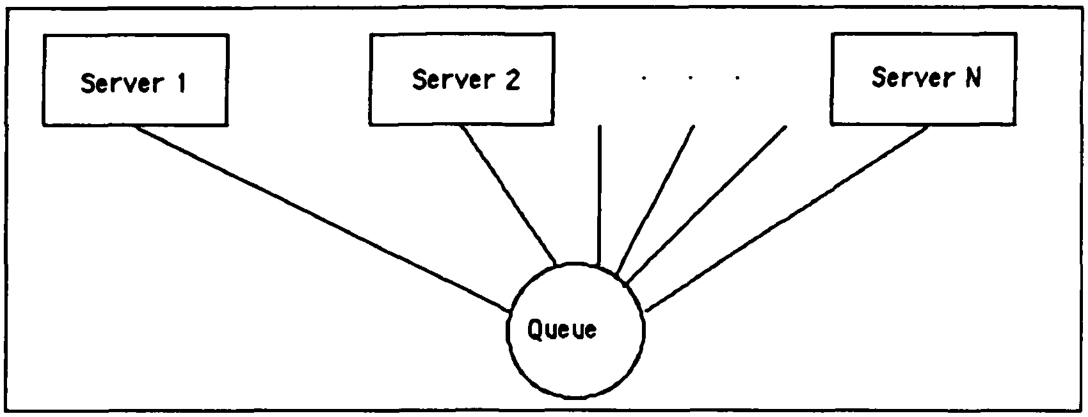


Figure 2. Single Uueueing System

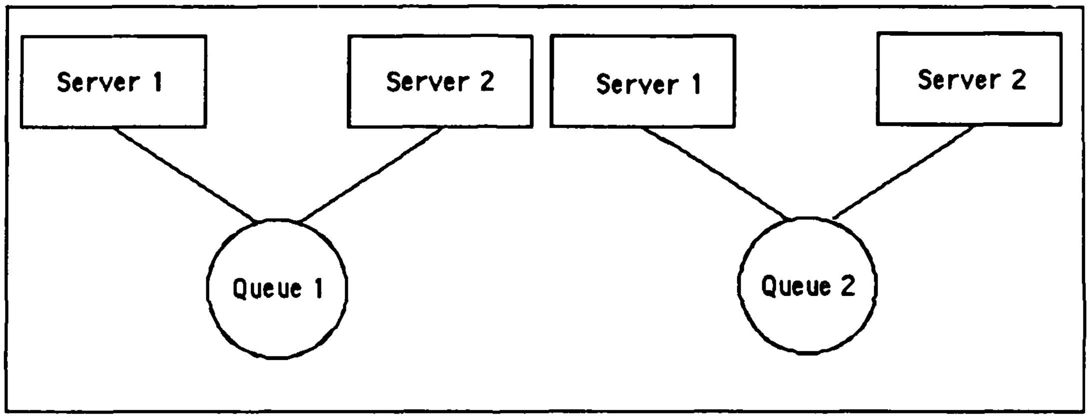


Figure 3. Multiple Queueing System

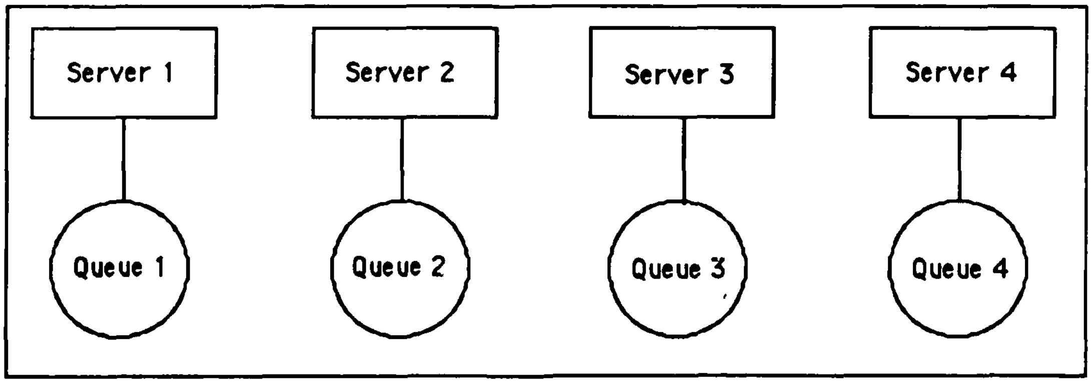


Figure 4. Multiple (one-to-one) Queueing System

presently in. The technique used to model assigning customers to queue lines in the model is described in Chapter III in the section on FORTRAN Subprograms.

Commissary managers call the checkout and queueing system the "front end" of the commissary. The model was developed so that the user can run the model with different front end configurations. For example, the model will simulate any of the three different types of queueing systems discussed above depending on the choices the user makes when executing the model.

Another important factor considered in the development of the model was the arrival of customers to the commissary. Clearly, the number of customers that arrive at a store will influence all of the operating aspects of the store. If large numbers of customers arrive during a given time period, then there will be congestion in the aisles and long queue lines. Subsequently, the average waiting time of customers in the queues will increase unless more cashiers are brought on duty.

For the purposes of this research, it was assumed that customer arrivals would follow a Poisson distribution. This is the case where arrivals to a queueing system occur completely randomly, but at a certain fixed mean rate, regardless of how many customers already are there so the size of the input source is infinite (7:526). Obviously there is not an infinite number of customers that shop at commissaries, but the number of customers is sufficiently

large so that the Poisson Process may be used to approximate the arrival process. In other words, the Poisson Process is an appropriate approximation with a large number of customers, each with a small probability of arriving at any given time. An intuitive way of describing a Poisson Process is that every time period of fixed length has the same chance of having an arrival regardless of when the preceding arrival occurred (7:537).

As discussed in the literature review, a commissary is a dynamic system with changing arrival rates, and therefore the commissary was not modeled as a steady-state system. To account for the dynamic nature of customer arrival rates, the model uses an algorithm that generates arrival times from a non-stationary Poisson process (2:99).

In summary, the methodology employed in developing the model was to focus on the flow of customers through a commissary. User-friendliness and generality were major considerations. The user interface consists of three input phases which are 1) general information such as number of parking spots and number of shopping carts; 2) front end information such as number of queue lines and number of cashiers ; and 3) customer arrival rates. A complete example of model execution which shows the menus displayed to the screen during user input is given in the user's guide in Appendix B.

Model Description

The model consists of the SLAM network and FORTRAN subroutines. Each of these is described below.

SLAM Network. The network capability of SLAM readily lends itself to model the flow of customers through a commissary as described above. A listing of the SLAM code is in Appendix A.

During interviews with AFCOMS management it was learned that each commissary store manager has the flexibility to set up the checkout lines as deemed to be best at that store. It was also learned that some stores use single queues while others use multiple queues. It was therefore necessary to design the SLAM network so that any configuration from one up to 35 queues could be modeled. The discrete event modeling capability of SLAM was employed to make the model general enough to represent these varied configurations (22:380-424). Some possible queue configurations were shown in Figures 2, 3, and 4.

Cashiers, baggers, parking spots, and shopping carts are all treated as resources. In this context, resources are simply a finite number of persons or objects that perform particular functions. For example, a cashier can only serve one customer at a time. If all cashiers are busy, then some customers may have to wait in line to be checked out. If all parking spots are taken, a customer may have to wait for someone to leave the parking lot before they can park their car. The user interface allows the setting of the resource

capacities before the simulation begins. The network also allows for either cash or check customers and for either regular shopping or express line customers. The user also is able to set the percentage of cash and check customers as well as the percentage of express line and regular customers.

FORTRAN Subprograms. Great flexibility was introduced into the simulation through the use of these subprograms. The three phases of user input are made possible through these modules. Each module is described below.

Subroutine Alloc. It is assumed that checkout service can begin only when a cashier and a bagger are available. This subroutine checks to see if both resources are available. If so, then one of each resource is seized so that service can begin for the first customer in line. If both resources are not available, service cannot begin.

Subroutine INTLC. This subroutine allows the user to set up the configuration for the commissary to be simulated. The user can either build a new configuration, use an old configuration, or exit the simulation. If a new configuration is desired, the user is prompted to input all general information, front end information, and arrival rate information. At all three phases of input, the user is asked to verify the correctness of the data and is given the opportunity to make corrections if needed. Once all data is verified, the information is written to a data file for possible future use. Next the user is asked to specify a file

name which will contain the output statistics from the model. Also, the user must specify how many runs of the model are to be made. The simulation begins execution at this point. If the user chooses to use an old configuration, the program prompts for the name of the data file the old configuration was stored in. As before, the user is asked to verify that the data is correct and is given the chance to make corrections if needed. Once the user verifies the data as correct, enters a file name for the output statistics, and specifies the number of runs desired, the simulation begins. If the user chooses not to continue, the exit option can be chosen and the program is terminated.

Subroutine Event. This subroutine contains twelve events which are called at various times during the simulation. A list of these events along with a brief description of each is given in Table I. Details of each event are given next.

The first event is called each time a customer finishes shopping and is ready to join a queue line. It is assumed that customers will try to choose the queue line which will offer them the shortest wait in line. A customer can observe the length of the queue lines and the number of cashiers servicing the queue lines and quickly see whether there is a better line to join. If no best alternative seems apparent, then the customer will join any line. The code in this event calculates the expected waiting time of a customer in each queue line and places the customer in the line with

Table I. List of Events

Event Number	Purpose
1	Places each customer in shortest checkout queue.
2	At fifteen minute intervals, calculates expected waiting time of customers in each queue line. Opens another checkstand and/or queue line if expected waiting time exceeds fifteen minutes. Closes a checkstand and/or queue line if expected waiting time is less than fifteen minutes and current arrival rate of customers is less than previous hour's arrival rate.
3	Prints message to screen to signal user that the commissary is closed.
4	Calculates the actual average waiting time of customers in all checkout queues.
5	Performs the same function as event 2 except only the express lines are considered.
6	Closes a checkstand after all customers in line at the time a queue line was closed have been serviced.
7	Changes the congestion factor to reflect low congestion in the store.
8	Changes the congestion factor to reflect no congestion in the store.
9	Changes the congestion factor to reflect medium congestion in the store.
10	Changes the congestion factor to high congestion in the store.
11	Assigns each regular customer the number of items bought.
12	Assigns each express customer the number of items bought.

the smallest expected wait time. The expected waiting time is calculated by dividing the number of customers already in the queue by the product of the number of cashiers servicing the queue and the mean service rate.

The second event is scheduled to execute every fifteen minutes and checks to see if the expected waiting time of customers in the queue lines exceeds fifteen minutes. If so, then an attempt is made to open another checkstand, that is, bring another cashier on duty. If another checkstand can be opened for the queue line, then a cashier is added and three baggers are also added. If the expected waiting time was greater than fifteen minutes but all checkstands for the queue line were already open, then an attempt is made to open another queue line. The user specifies the maximum number of queue lines and the maximum number of checkstands that service each queue during the input of the front end information as described in the Methodology section.

The next segment of event two checks to see if a checkstand can be closed. This first check is made after the simulation has been running for one hour of simulated time and every fifteen minutes thereafter. Expected waiting time is again calculated. If the waiting time is less than fifteen minutes and the current customer arrival rate is smaller than the previous hour's customer arrival rate, then a checkstand will be closed and the cashier and bagger resources are altered accordingly. If the special case exists when there is only one checkstand servicing a queue line, then provision is

made to stop putting customers in that queue line and also to service the customers already in the line. The last segment of code in event two calculates the total number of cashiers on duty at fifteen minute intervals. This value is written to a data file so that the user can latter compare the number of cashiers on duty with the actual waiting time of customers which is calculated in event four.

The decision to check the state of the system, that is expected waiting times and congestion thresholds, at fifteen minute intervals was the choice of the author during model development and validation. Provision is made in the model so that the user can check the state of the system at time intervals other than fifteen minutes.

The fifth event performs the same function as event two except that only the express lines are closed or opened.

The third event is executed after the simulation has been running for the amount of time the commissary was to be open. A message is displayed to the screen of the computer to signal the user that the door of the commissary was closed and no more customers will arrive. The customers currently in the store will all be serviced and then the simulation is terminated.

During validation of the model, it was discovered that after the arrivals of customers ended, it took the model too long to process the remaining customers in the system. After discussion with the author's advisor and local commissary officials, it was decided that the reasons for the lengthy

clearing-out period after closing was that the model made no provision for the type of shoppers entering the store nor the change in service rates of the cashiers. How the model was corrected to handle these two situations is discussed next.

First, it was believed that the shoppers arriving at the commissary in the hour before closing were shopping for fewer items than shoppers throughout the rest of the day. This conjecture was substantiated through a closer review of the cash register tapes. Also, the author observed while collecting arrival rate data at the Wright-Patterson store that customers arriving within an hour of closing time were buying fewer items as well as shopping faster. Therefore, the model was changed to predict a different shopping time for customers arriving to the store within 45 minutes of closing time.

Second, commissary managers at the Wright-Patterson store feel that cashiers' service rate speeds up after closing time. This is intuitively reasonable since the cashiers know that as soon as all customers have been serviced that the work day will be over. The model was changed to reflect a faster checkout time after closing.

Event four calculates the average waiting time of customers in all queues. This is the actual waiting time of all the customers who left the system in the last fifteen minutes. Every 15 minutes of simulated time, the average waiting time of customers, the hour of operation, the number of customers in the store, and the number of cashiers on duty

are all printed to the screen so that the user can see the status of the system.

Event six is scheduled to execute whenever a queue line was closed by event two and there were customers in the line at that time. A check is made to determine if all customers in the queue have been serviced. If so, then the cashier and bagger resources are freed to be used elsewhere. If all customers have not been serviced, then the event is scheduled to occur again after five minutes. After all customers have been serviced the resources can be altered.

Events seven, nine, and ten are called to change the value of the congestion factor whenever the number of customers in the commissary has crossed thresholds specified by the user. Intuitively, a threshold is the maximum number of customers that can be in the store before the store becomes so congested that customers' shopping times will be increased. The three thresholds in the model represent low, medium, and high congestion. For example, if a commissary can handle 200 customers before low congestion is present, then the threshold would be 200. The value of the congestion factor is 1.0 when the store is in a non-congested state. If any of the three thresholds is exceeded, the congestion factor is increased accordingly to reflect that the store is in a low, medium, or high congested state. For example, if a shopper would have spent 60 minutes shopping in a non-congested commissary but spent 75 minutes because of congestion, the additional 15 minutes represents a 25 percent

increase in shopping time and is reflected by increasing the congestion factor to 1.25.

Event eight is called whenever the number of customers in the store indicates that congestion is no longer present. When called, event eight simply resets the value of the congestion factor to 1.0 to reflect that congestion is not increasing shopping times of customers.

At fifteen minute intervals, the model computes the total number of customers in the store. If the threshold is exceeded, event seven is called. If the number of customers in the store drops below the threshold, then event eight is called. Details of how the thresholds were determined will be given in the next chapter.

Event eleven assigns each regular customer the number of items that customer buys. The number of items assigned to each customer is determined from a probability distribution which the user must specify. Details of determining this probability distribution are given in Chapter IV. Attribute three of each entity is set to one if less than 100 items are bought or two if more than 100 items are bought. The customer will require one shopping cart if attribute three is equal to one or two carts if attribute three is equal to two.

Event twelve assigns each express customer the number of items that customer buys.

<u>Subroutine Error</u>. If the user makes an invalid response to a prompt, this subroutine is called to signal the user so that the correct response can be entered.

__Function Userf__. This function performs four operations. A list of the four functions along with a brief description of each is given in Table II.

Table II. List of Functions

__Function Number__	__Purpose__
1	Generates interarrival times of customers.
2	Calculates the shopping time of each customer.
3	Calculates the checkout time of each regular customer.
4	Calculates the checkout time of each express customer.

The first operation is the generation of interarrival times of customers. This portion of the code was written by Captain John Hertz. Captain Hertz used Cinlar's algorithm for generating non-stationary Poisson arrivals which was described in the methodology section above (2:99).

The second operation calculates the shopping time of each customer based on the number of items the customer buys and the amount of congestion in the store. The equation that predicts the shopping time is a result of linear regression and is as follows:

$$y = (b0 + b1x + e)C \tag{1}$$

where : b0 is the y-intercept
b1 is the slope
x is the number of items the customer buys
e is the error term which introduces a random element into shopping times
c is the congestion factor

The data used to perform the regression was obtained from a sample of commissary shoppers who tracked their shopping times and the number of items they purchased. Chapter IV discusses this data collection and regression in detail. The congestion factor was modeled using the approach suggested by Gordon of letting congestion be a function of the number of customers in the store (5:8).

The value of the congestion factor is set in Subroutine Event. If no congestion is present, then the factor is 1.0 and no increase in shopping time is shown. If congestion is present, the factor is greater than 1.0 and an increase in shopping time will be reflected. For example, if a shopper's predicted shopping time is 60 minutes and the congestion factor is 1.0, then the total shopping time will still be 60 minutes. However, if congestion is present and the congestion factor is, say 1.25, then the total shopping time will be the predicted time multiplied by 1.25 or 75 minutes.

The third function calculates the checkout time of each customer based on the number of items the customer buys. As for the shopping times, the equation that predicts the checkout time was obtained through regression and is as follows:

$$y = (b0 + b1x + e) \tag{2}$$

where : b0 is the y-intercept
b1 is the slope
x is the number of items the customer buys
e is the error term which introduces a random element into the checkout times

Note that the congestion factor is not present. Chapter IV discusses the results of the regression.

The fourth function calculates the checkout time for express customers based on the number of items they bought. Equation 1 is also used for this computation.

<u>Subroutine Arvl</u>. The third phase of user input is obtained in this subroutine. The user is prompted to enter the arrival rates for each hour of operation. After the rates are input, the user is asked to verify they are correct and to make corrections if necessary.

As discussed in the literature review, customer arrival rates to commissaries are non-stationary. For example, during data collection at the Wright-Patterson AFB Commissary, it was observed that the arrival pattern varied throughout the day. Figure 5 shows a possible arrival pattern.

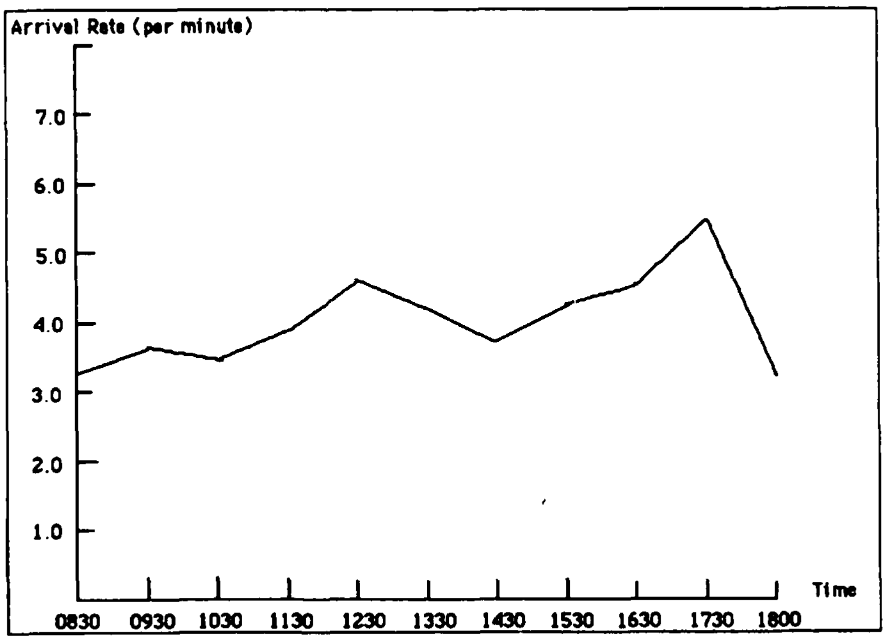


Figure 5. Example of Customer Arrival Pattern

Subroutine Otput. Since the model was developed to simulate many different configurations of commissaries, it was necessary to tailor the output statistics so that only the items of interest would be output. This was accomplished by calling the SLAM output functions from this subroutine. The output statistics are written to the file specified by the user as described in the section on Subroutine Intlc.

This subroutine also tracks the number of runs of the model that have been completed. As discussed in the section on Subroutine Intlc, the user must specify the number of runs desired. When all runs have been completed, Subroutine Otput calculates mean estimates and 95% confidence intervals for the average waiting times of customers in the checkout queues. These estimates and confidence intervals are calculated for each 15 minute interval during open hours. After closing, the model calculates a mean estimate and 95% confidence interval for waiting times from when the store closes until all customers have been serviced. Also, the average number of cashiers on duty and the minimum and maximum number of cashiers on duty for each 15 minute interval during the day is calculated. A sample of this output file is given in Chapter V.

Summary

This chapter has discussed the development of the model to include the methodology employed and a description of the model. The next chapter covers data collection and analysis.

IV. Data Collection and Analysis

Overview

A subsidiary objective of this research effort was to develop guidelines for collecting the data needed to run the simulation model. These guidelines were developed as the researcher gathered the data at the Wright-Patterson AFB Commissary.

It was decided to collect the data for a specific day of the week. There were two reasons for collecting the data in this manner. First, since the model simulates one day's operation of a commissary, it seemed reasonable to collect data for a specific day so that after collection and analysis, the model could be run to see if the predicted throughput of the simulation was close to the actual throughput of the store on that day. If the model correctly predicted throughput for one day, one would have a high degree of confidence that the model would work as well for any day of the week, given that the data was correctly collected and analyzed. Secondly, it would not be possible to collect and analyze the huge amounts of daily data generated at the Wright-Patterson AFB Commissary during the time allotted to work on this research project. Therefore, the researcher arbitrarily chose Friday as the day for which data would be collected.

The areas of interest during the data collection and analysis were as follows : 1) customer arrival rate;

2) number of items a customer bought; 3) customer shopping time; 4) congestion factors; 5) checkout time; 6) percentage of cash versus check customers; and 7) percentage of express versus regular customers. Each of these will be discussed in the order listed. After data collection and analysis is discussed, guidelines will be given that will help future users of the simulation model collect the appropriate data.

Arrival Rate

Customer arrival rates to the Wright-Patterson AFB Commissary were observed for the three Fridays between 2 October 1987 and 23 October 1987. The author arrived at the store before opening at 0830 hours and stayed until closing at 1800 hours. The number of customers waiting at the front door when the store opened was noted. The number of customers arriving at the store were counted on an hourly basis. In Chapter III, justification was given for assuming that arrivals to the store would follow a Poisson Process. The Poisson probability distribution is of the form

$$p(y) = \frac{\text{lambda}^{y}}{y!} \exp(-\text{lambda}) \qquad (3)$$

where lambda is the single parameter of the distribution and is the number of arrivals expected during a single time interval (16:93). Since it was decided that the arrivals would be considered as a Poisson process, it was easy to obtain the arrival rate per minute by simply dividing the number of arrivals each hour by 60.0. The conversion of

the hourly number of arrivals to arrivals per minute is necessary since the simulation model generates customer arrivals per minute.

Number of Items Bought

The service scheduler, Mrs. Helen Jennings, at the Wright-Patterson store provided the author with cash register detail tapes which contained the following information : 1) time customer began checkout; 2) time customer ended checkout; 3) whether paid by cash or check; and 4) a listing of items purchased. All of this information was needed in the data collection process. As for all other data collected, Friday was the only day of the week considered.

Care was taken to make sure that the samples selected from the detail tapes were representative of all the hours of operation. A total of 454 transactions were reviewed. Of these, 343 were from the regular checkout registers and 111 were from the express line cash registers. This involved going through the detail tapes manually and counting the number of items purchased by each customer. The 343 data points from the regular checkout registers were then analyzed with the AID software package by Pritsker and Associates. The Kolmogorov-Smirnov goodness of fit test showed that at the .2 level of significance, the distribution of the number of items could be represented as Gamma with alpha equal 3.2 and beta equal 23.1. A graphical representation of

the Gamma distribution obtained by using the AID software package is shown in Figure 6.

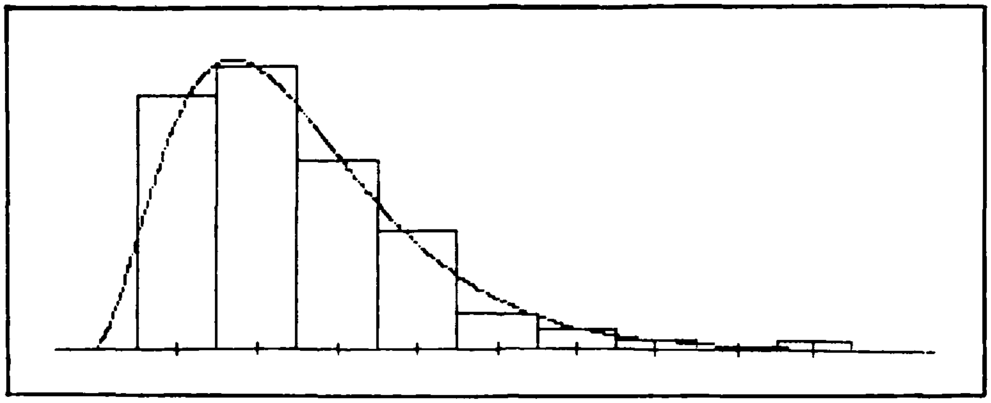


Figure 6. Gamma Probability Distribution

It was also necessary to determine a distribution for the number of items express customers bought. The 111 data points from the express cash register tapes were also analyzed with the AID software package. The Kolmogorov-Smirnov (K-S) goodness of fit test showed that at the .2 significance level the data could be represented as Uniform with mean 7.5 and variance 16.3. Figure 7 shows a graph of the Uniform distribution as obtained using the AID software package.

In both of the goodness of fit tests above, the null hypothesis was that the data in the sample were from a specified distribution. The alternative hypothesis was that the data were not from the specified distribution. There are

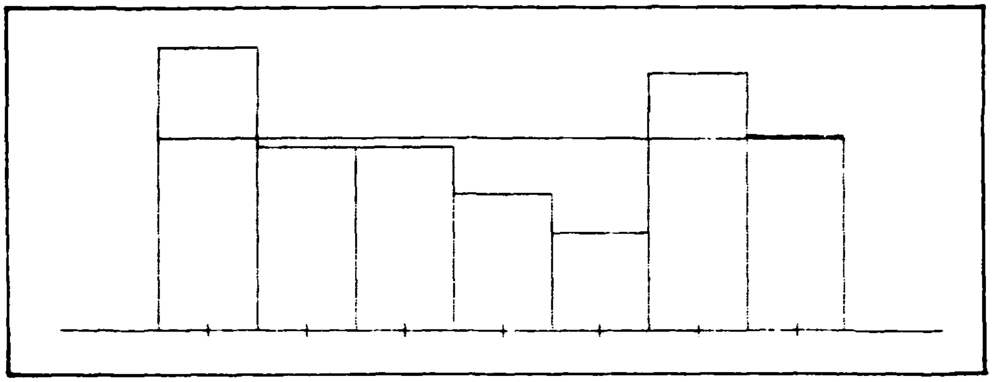


Figure 7. Uniform Probability Distribution

four levels of significance which the test can be conducted at : .01, .05, .10, and .20. As the level of significance gets larger, it becomes easier to reject the null hypothesis. Therefore, since both of the tests were conducted at the .2 level and both samples had a large number of data points, the fact that neither was rejected indicates very strongly that the two sets of data are from the specified distributions.

It should be noted that in the course of this research, the author learned that the AID Software uses the standard Kolmogorov-Smirnov table to test for the goodness of fit. Since the parameters of the distributions must be estimated from the sample data,the level of significance is diminished. The test thus becomes conservative. However, the author was able to obtain a modified K-S table for the Gamma distribution (17). Computations showed that the data still fit at the .2 level. No modified table was found for the Uniform distribution, therefore the author concluded that the

K-S test for the Uniform distribution was conservative, that is, the level of significance is less than .2. However, the Uniform distribution was still considered to be a good approximation for the number of items express customers buy.

The determination of a distribution for the number of items a customer buys was very important since SLAM has intrinsic functions which will generate random deviates from probability distributions. The simulation model thus uses a random Gamma deviate as the number of items for each regular customer in the simulation and a Uniform deviate as the number of items for each express customer. Since the Gamma and Uniform distributions are continuous, the deviates from each distribution are real numbers. Obviously customers cannot buy fractional parts of items, so the simulation model simply takes the integer part of these numbers as the number of items bought by customers. Once the number of items is determined for a customer, then the amount of time the customer spends shopping and the amount of time it takes to check the customer out can be determined by other statistical models which are developed in the next two sections.

Customer Shopping Time

Data collection for customer shopping times was accomplished by soliciting volunteers from the researcher's classmates and neighbors who shop at the Wright-Patterson AFB Commissary. These volunteers were asked to track the actual time they spent shopping, that is the time from when they

picked up a shopping cart until the time they joined a queue line to wait for checkout. The time spent in the waiting line was not measured. The volunteers also provided the number of items which they had picked up while shopping.

It was hypothesized that the amount of time a customer spends to shop depends on the number of items the customer picks up. That is, it was believed there is a relationship between the time spent shopping and the number of items bought. For this reason, a regression model was chosen to fit the data provided by the volunteers.

The first functional form chosen to fit the data was the simple linear regression model. The model can be formally stated as follows (18:31) :

$$Yi = B0 + B1Xi + Ei \tag{4}$$

where :

Yi is the dependent or response variable in the i-th trial. For this model, Yi is the shopping time.

B0 and B1 are the regression coefficient parameters with B0 as the y-intercept and B1 is the slope of the regression line.

Xi is a known constant. For this model, Xi is the number of items purchased by the i-th customer and is determined by a random draw from a Gamma distribution.

Ei is the random error term which is assumed to be distributed Normally with mean zero and constant variance.

The model is said to be simple, linear in the parameters, and linear in the independent variable (18:31).

Since B0 and B1 were unknown, they had to be estimated from the data collected. The estimates , b0 and b1, for these parameters were obtained through the regression procedure in the SAS System (23). This procedure uses the method of least squares to obtain estimates for regression coefficients. The Gauss-Markov theorem states that the least squares estimators, b0 and b1, are unbiased and have minimum variance among all unbiased estimators. In other words, since the estimators are unbiased, they do not tend to overestimate or underestimate systematically. Also, since they have minimum variance, these least squares estimators are more precise than any other unbiased estimators which are linear functions of the observations Y1,...,Yn (18:39).

Since the error terms are normally distributed with mean of zero and constant variance, the variance can be estimated by the Mean Square Error (MSE) given by the regression procedure in SAS. The fitted model thus becomes :

$$\text{Ypred} = b0 + b1X1 + e \quad (5)$$

where:

Ypred is the predicted value of the response variable.

b0 is the estimate of the y-intercept.

b1 is the estimate of the slope.

X is the number of items the customer bought.

e is the error term, that is a random normal deviate from a normal distribution with mean zero and variance equal to MSE.

SAS was used to fit the data to the above model. The null hypothesis was that there was no linear relationship between the number of items a customer buys and the amount of time spent to shop for those items. The alternative hypothesis was that there was a linear relationship. The Analysis of Variance (ANOVA) table in Table III shows the results of the regression.

The F-value was 62.583 which indicates that the null hypothesis should be rejected in favor of the alternative hypothesis, that is, there is a linear relationship between the time spent to shop and the number of items bought. It is interesting that the R-Square value is about 70% which means that this very simple model accounts for 70% of the variability in shopping times.

Table III. ANOVA Table for results of regression for shopping time.

Analysis of Variance

Source	Degrees of Freedom	Sum Of Squares	Mean Square	F Value
Model	1	8106.34	8106.34	62.583
Error	26	3367.76	129.53	
Total	27	11474.10		

Root MSE 11.38 R-Square .7065

Parameter Estimates

Variable	Parameter Estimate	DF	Standard Error
Intercept	13.8	1	4.25
Numitems	.4	1	.05

Model adequacy was checked by examining the residuals. A plot of the residuals versus the predicted values showed that the variance was constant. A normal probability plot of the residuals also showed no serious departure from normality. Therefore the model was assumed to be adequate (18:111-122).

The simulation model thus uses the above regression model to predict the amount of time each customer will spend shopping based on the number of items the customer buys. As described in the previous section, the number of items a customer buys, denoted by X in the regression model, is determined from a random draw from a Gamma distribution. Figure 8 is a graphical representation of the composite model used to calculate shopping times.

An example of how this composite model works is now given. Suppose that the value of a random deviate drawn from the Gamma distribution shown in the top portion of Figure 8 is 70. This value is then assigned to the X1 variable in the regression equation shown in the bottom portion of the same figure. The error term in Equation 5 is also a random deviate, but from a Normal distribution with mean zero and constant variance. If the value of the random deviate drawn for e is 4.2, then the value of y would be 46.0, which means that this customer's shopping time based on the number of items would be 46.0 minutes.

It should be noted that a model with a quadratic term was also used to fit the shopping time data. The null hypothesis was that there was no quadratic relationship

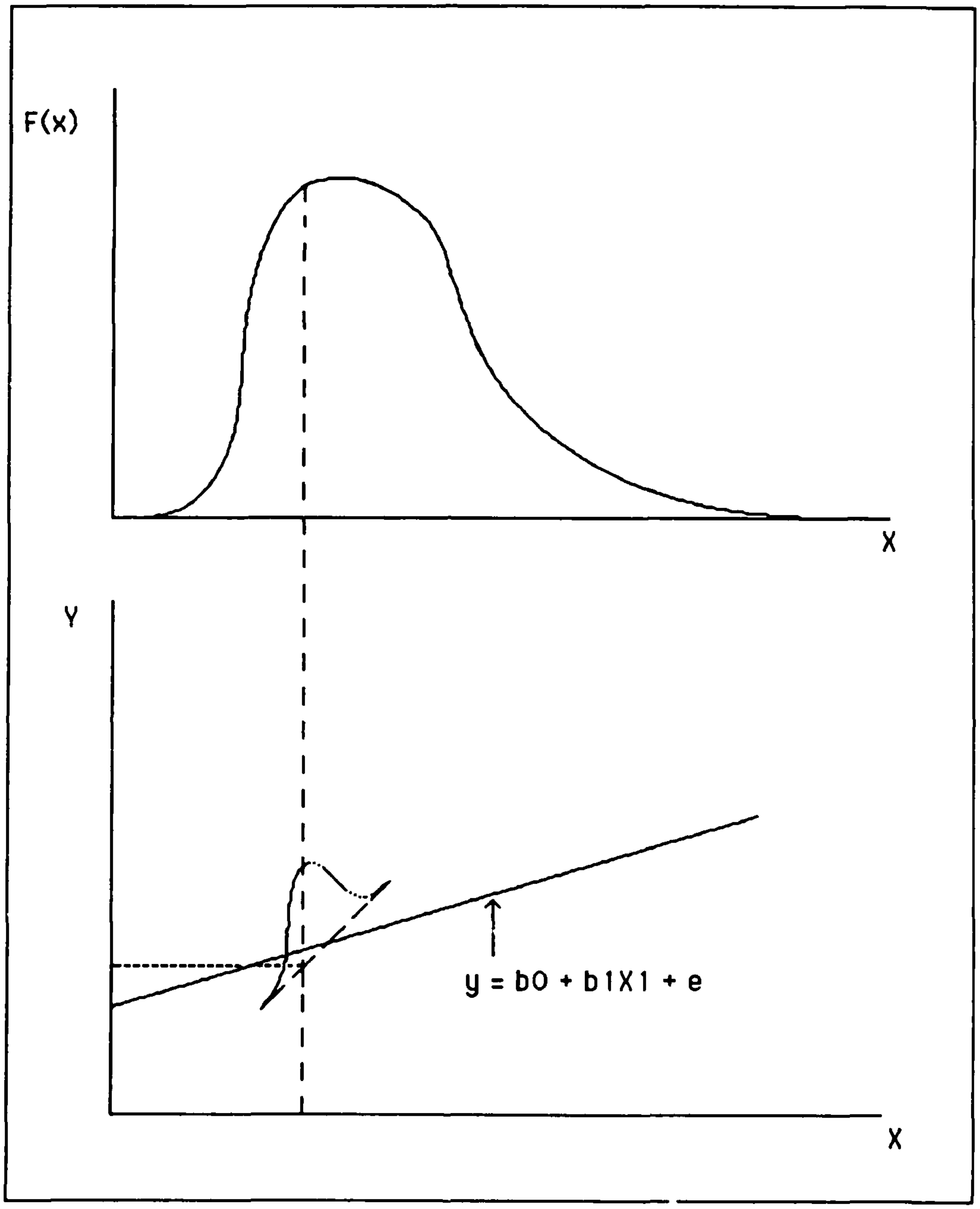


Figure 8. Composite Model Used for Predicting Shopping and Checkout Times

between shopping time and the number of items bought, and the alternative hypothesis was that there was a quadratic relationship. This model specification included a quadratic term since it was believed that shopping time in relation to the number of items picked up may not follow a strictly linear relationship. The model chosen was as follows:

$$Yi = B0 + B1X1 + B2X1^2 + Ei \qquad (6)$$

where :

Yi is the dependent or response variable in the i-th trial. For this model, Yi is the shopping time.

B0, B1, and B2 are the regression coefficient parameters.

Xi is a known constant. For this model, Xi is the number of items purchased by the i-th customer and is determined by a random draw from a Gamma distribution.

$X1^2$ is the quadratic term

Ei is the random error term which is assumed to be distributed Normally with mean zero and constant variance.

SAS was used to fit the data to this model. The results of the analysis showed that the quadratic term was insignificant, that is, the null hypothesis that there is no quadratic relationship between shopping time and the number of items should not be rejected.

Congestion is another factor which affects shopping time and is discussed in the next section.

Congestion Factor

As discussed in Chapter III, the other factor that influences a customer's shopping time is congestion in the store. The congestion factor was viewed as a percentage increase over the normal time a customer would spend shopping for a given number of items. Details of how the simulation model changes the congestion factor to reflect a congested state were outlined in Chapter III.

The simulation model predicts each customer's shopping time based on the number of items bought and the congestion factor. The equation that predicts total shopping time is then as follows:

$$Ypred = (b0 + b1X + e)C \tag{7}$$

where :

Ypred is the predicted shopping time

b0 is the estimate of the y-intercept

b1 is the estimate of the slope

X is the number of items bought, determined by a random draw from a Gamma distribution

e is the estimate of the error term, MSE

C is the congestion factor

In order to determine what the congestion factor thresholds were, it was necessary to obtain the expert opinion of the Commissary Store Manager at Wright-Patterson AFB, Mr. Don Johnson. Mr. Johnson was asked to estimate, based on his own knowledge and experience, how many people

the Wright-Patterson AFB Commissary could support before congestion starts to increase shopping times by ten, twenty, and forty percent. Mr. Johnson's estimates were 350, 400, and 500 respectively (10). These values were used in the model as thresholds to change the congestion factor.

The days on which data was collected at the Wright-Patterson AFB Commissary, congestion was usually in the low or medium state. However, on 2 October, the congestion factor was high since it was observed that for several hours that all 450 shopping carts were being used.

Checkout Time

The next data collected was customer checkout times. This information was easily obtained from the cash register detail tapes. At the same time the tapes were reviewed to obtain the 343 observations of number of items purchased, the time it took to checkout those 343 customers was also collected. Thus the data set consisted of 343 observations of checkout times versus number of items purchased.

Just as the time to shop was dependent on the number of items a customer buys, it was believed that the time to checkout would also be dependent on the number of items. Therefore, the same simple linear regression model that was used to fit the shopping time data was chosen to fit the checkout time data. SAS was again used to fit the data and to test the null hypothesis that there was no linear relationship between the number of items and checkout time

versus the alternative hypothesis that there was a linear relationship.

The data were fit to the model and the results are shown in Table IV.

Table IV. ANOVA table for results of regression for checkout times.

Analysis of Variance

Source	Degrees of Freedom	Sum Of Squares	Mean Square	F Value
Model	1	1880.08	1880.08	559.748
Error	341	1145.35	3.35	
Total	342	3035.43		

Root MSE 1.83 R-Square .6214

Parameter Estimates

Variable	Parameter Estimate	DF	Standard Error
Intercept	1.43	1	.199
Numitems	.05	1	.002

The F-value was 559.748 which indicates that the null hypothesis should be rejected in favor of the alternative hypothesis. That is, there is a linear relationship between checkout time and number of items. Therefore the simulation model uses the following equation to predict each customer's checkout time based on the number of items the customer bought :

$$Ypred = b0 + b1X + e \tag{8}$$

where :

Ypred is the predicted checkout time.

b0 is the estimate for the y-intercept of the regression line.

b1 the estimate of the slope of the regression line.

e is the error term with mean zero and the estimate of the variance is MSE

Model adequacy was checked through residual analysis. The variance appeared to be constant and the residuals appeared to be normally distributed.

Cash versus Check Customers

This data was collected from the detail tapes at the same time the number of items and checkout times were obtained. Of the 454 observations, 10% were cash customers and the rest were check writing customers. These percentages were used in the simulation model.

Express versus Regular Customers

This information was obtained from the detail tapes of the cash registers used for the express lanes. For the Fridays concerned, 15% of the total customers were express customers.

Data Collection Guidelines

During data collection at the Wright-Patterson AFB Commissary, guidelines for data collection by potential users of the simulation model were developed. Guidelines are given for each of the seven areas for which data must be collected.

First, customer arrival rates can be deterr ned by simply counting the number of customers that arrive at the front door each hour. This could be easily accomplished by the employee who checks identification cards at the door. Since the simulation model uses arrivals per minute, it will be necessary to divide each hour's number of arrivals by 60.0 to obtain the correct rate to input to the model. A better approach would be to count arrivals to the store in 15 minute intervals and then convert to arrivals per minute by dividing the number of arrivals during each interval by 15.0. This approach would provide a more detailed model of the time-varying arrival rates.

The next information the simulation model needs is a probability distribution of the number of items customers buy. This distribution can be determined from a sample of the customers who processed through the commissary in previous weeks on the day the user is interested in simulating. For example, if the user wants to simulate a day's operation for an upcoming Thursday, the sample should be taken from previous Thursdays' cash register detail tapes. Careful attention should be paid to be sure that all the hours of the operating day are included in the sample. A general rule for sample size is to collect between 20 and 100 data points and then apply a goodness of fit test such as the Kolmogorov-Smirnov test (17). Once the data is collected, a software package such as AID can be used to fit the data to a probability distribution. Once the distribution is known, the

simulation model can be coded so that customers will be assigned a number of items bought. It should be noted that the data should be collected for regular customers and express customers. That is, distributions for both types of customers must be determined.

To determine customer shopping times, commissary managers could solicit volunteers from among commissary patrons to track the actual time they spend shopping. The shopping time includes the time from when the customer picks up a cart until they join a queue line to wait for checkout. Time in the queue line is not to be included in the shopping time. Once a sample of 20-50 customers' shopping times has been collected, a statistical software package such as SAS can be used to fit simple linear regression model. The equation obtained from the regression will be used in the simulation model to predict the shopping time of each customer based on the number of items the customer buys. Recall that the number of items bought by a customer is determined from the probability distribution found above.

The other factor that affects shopping time is congestion in the store. To obtain the information the simulation model needs to change the congestion factor, the commissary manager of the store under study should be asked to estimate the thresholds at which congestion will cause a 10%, 20%, and 40% increases in customers' shopping times. The estimates should be in numbers of customers. For example, at

the Wright-Patterson AFB Commissary, the manager estimated that when there are 300, 350, and 400 customers in the store that shopping times increase by 10%, 20%, and 40% respectively.

An alternative way to measure the effect of congestion would be to measure customers' shopping times during the periods that congestion is present and the actual number of people in the store is known. Determination of the number of shoppers in the store could be accomplished by tracking the number of shopping carts in use during specified time periods. The shopping times of customers could then be compared to shopping times when congestion is not present to determine the congestion thresholds and percentage increase of shopping time. This approach suggests that a regression model with two variables--number of items and number of customers in the store--may be an appropriate functional form with which to model shopping times.

Checkout times for customers are also easily obtained from the cash register detail tapes. As the tapes are being reviewed to determine the number of items customers bought, the beginning and ending time of each customer's checkout should be noted. This data can be fit to a simple linear regression model by a statistical software package such as SAS in the same manner that the customer shopping times were fit. The equation obtained through the regression will be used by the simulation model to predict each customer's checkout time based on the number of items bought.

To determine the percentage of cash customers, the detail tapes should be reviewed at the same time the number of items and checkout times are being collected. Each customer's order shows if they paid in cash or by check. The percentage of cash customers will be the number of cash customers divided by the total number of customers' orders reviewed in the detail tapes.

Finally, to determine the percentage of express line customers, the detail tapes of the express line registers should be reviewed to see how many express line customers were serviced for that day. This number should be divided by the total number of customers who were serviced in the store that day to obtain the percentage of express line customers.

Details of how to code the simulation model to reflect the correct probability distribution of number of items, the predictor equations for shopping times and checkout times, congestion thresholds, and percentages of cash customers and express customers is given in the User's Guide in Appendix B.

Summary

This chapter has discussed data collection and analysis. The methods used to collect and analyze the data necessary to run the simulation model were covered in detail. Guidelines for data collection and analysis were also given for potential users of the model. The next chapter discusses model verification, validation, and output.

V. Model Verification, Validation, and Output

Overview

After the simulation model had been programmed in SLAM and FORTRAN, the performance of the model was evaluated. Evaluation was accomplished through verification and validation of the model. This chapter discusses the verification and validation procedures used. Also, a sample of model output is presented along with comments on how to interpret the output.

Model Verification

"The verification task consists of determining that the translated model executes on the computer as the modeler intended " (22:12). Verification can be accomplished by manually checking the calculations the model performs. The model developed during this research effort was verified by the author. In addition, Captain Theodore P. Lewis, a classmate of the author, verified that the model performs as intended.

The model is general in that it can simulate many different configurations of commissary stores. For example, the model can represent stores with up to 35 checkout lines. Several test runs of the model were conducted to see if it would perform correctly under varying configurations and extreme conditions. An example of an extreme condition is a drastic jump or decline in arrival rates to the store. In this context, perform correctly means the model processes

entities through the system and produces output statistics as intended. For example, the model should respond to increasing customer arrival rates by opening more checkout stands and/or queue lines. Conversely, if arrival rates decline, the model should close some checkout stands and/or queue lines. Table V shows some of the configurations the model was tested with.

Table V. Verification Test Runs

Number of Queue Lines	Number of Checkout Stands
1	20
2	20
10	20
35	35

In the test runs conducted, the model performed as intended by opening and closing queue lines and checkout stands as arrival rates increased or decreased.

Another verification check made was the manual calculation of the 95% confidence intervals for the mean estimates of customer waiting times. It was determined from these manual checks that the model is correctly computing the confidence intervals.

Model Validation

" The validation task consists of determining that the simulation model is a reasonable representation of the system" (22:12-13). Pritsker goes on to say

> In simulation models, there is a correspondence between the model elements and system elements. Hence, testing for reasonableness involves a comparison of model and

system structure and comparisons of the number of times elemental decisions or subsystem tasks are performed [22:13].

Validation of the model was therefore accomplished in in two steps : 1) by comparing model output with expert knowledge of commissary management regarding performance behavior of commissaries and 2) by comparing model output with past output of the Wright-Patterson AFB Commissary. The first step is structural validation and the second step is output validation. These two steps are discussed next.

Structural Validation. Early in the model development stage, AFCOMS officials were contacted to determine the correct structure of a commissary system. Although commissaries vary in size and physical layout, all stores possess the basic structure described in Chapter III and depicted in Figure 1. This basic structure was validated through interviews with AFCOMS officials (9; 11; 20). Additionally, Major Solheim and Mr. Don Johnson at the Wright-Patterson AFB Commissary were interviewed and they also considered the structure of the model to be valid (10 ; 24).

Output Validation. Validation of output from the simulation model was done by comparing actual output of the Wright-Patterson AFB Commissary with model output. Data was collected and analyzed as described in Chapter IV. The model was run using this data as inputs.

On the days that arrival rate data was collected at the Wright-Patterson store, a log was kept on the the total

number of checkout stands that were open. This log was updated at 15 minute intervals. Also, the approximate number of shopping carts available to customers was observed. If the simulation model showed utilization of cashiers and shopping carts similar to the actual utilization observed in the store, then it would be reasonable to say the model is valid.

Another output of the model which is easy to compare to actual system output is the number of customers processed through the commissary during one day's operation. It was decided that since data was available for the number of shopping carts available, the number of cashiers on duty, and the number of customers processed through the store, the simulation model output would be compared to the actual system output for the three Fridays on which data was collected. It was expected the model's output would closely follow the output of the actual system for all three days if the model was valid.

The model was replicated five times and the results are shown in Tables VI and VII. From Table VI it is seen that the model output compares favorably with the actual commissary output on all three days. On October 2, 1987 for example, the actual throughput of customers was 2612 compared to an average of 2571 over the five runs of the model. This 1.5% difference is negligible and is probably due to error in counting customer arrival rates. On the same day, the model shows that all the shopping carts were being used while the actual system was observed to be saturated. That is, customers

had to wait for carts most of the day which means all 450 carts were being used.

Table VI. Model output versus System Output of number of customers and shopping carts utilized.

Date	Number of Customers		Maximum Number of Carts Utilized	
	System	Model	System	Model
2 Oct 87	2612	2571	450	450
9 Oct 87	2438	2371	425	420
23 Oct 87	2248	2201	400	364

It should be kept in mind that the model changes the number of cashiers on duty based on a decision rule which requires another checkout stand to be opened if customers' expected waiting times in line will exceed 15 minutes. Therefore, the model is dynamic and changes the number of cashiers on duty as the state of the system changes. As Table VII shows, during the first three hours of operation, the model output and the system status show approximately the same number of cashiers on duty. The apparent discrepancy between the actual system output and the model output in Table VII after the first three hours of operation can be explained by the dynamic characteristic of the model. By changing the number of cashiers on duty to prevent customers from waiting in line more than 15 minutes, the model is in effect showing how many cashiers should be on duty to meet the AFCOMS objective that no customers wait in line more that 15 minutes. The reason the actual system had fewer cashiers on duty is that no additional cashiers were available for

Table VII. Model Output versus System status of number of checkout stands open

Date	Time	Number of checkstands open System	Model
2 oct 87	0830	3	4
	0845	7	4
	0900	8	6
	0915	9	8
	0930	9	10
	0945	10	12
	1000	15	14
	1015	15	16
	1030	15	18
	1045	17	20
	1100	18	20
	1115	18	20
	1130	17	20
	1145	17	20
	1200	15	20
	1215	16	20
	1230	15	20
	1245	15	20
	1300	16	20
	1315	16	20
	1330	16	20
	1345	16	20
	1400	17	20
	1415	18	20
	1430	19	20
	1445	17	20
	1500	16	20
	1515	14	20
	1530	13	20
	1545	10	20
	1600	10	20
	1615	10	20
	1630	9	20
	1645	13	20
	1700	16	20
	1715	17	20
	1730	16	20
	1745	16	20
	1800	16	12

duty. It is very likely then that the model could show more cashiers on duty than were actually observed to be working.

Based on the results of the structural and output analysis presented in this section, the model appears to be a

valid or reasonable representation of a commissary. In the next section, an example of the output produced by the simulation model is given.

Model Output

As described in the literature review, simulation is experimental and each run of a model is an experiment. It must be emphasized that the modeler must determine the number of runs to make. Some discussion of output analysis is presented before the example of output is given.

Output Analysis. The performance measures of interest in this simulation model are average waiting times of customers in checkout lines and the number of cashiers on duty. Simulation output is a sequence of random variables representing the performance measures of the system. The objective in output analysis is to summarize the output by placing a confidence interval about the mean of each performance measure. Since the commissary simulation model is a terminating simulation, that is it ends when all customers have been serviced, the method chosen to obtain the mean estimates and confidence intervals was independent replications (14).

The model computes the average waiting times of customers at fifteen minute intervals. Suppose the model is run five times. The model will compute five observations of waiting time for each fifteen minute interval. For example, for the 15 minute interval between 0830 and 0845, an average

waiting time is determined for each run of the model. Using these five observations for this interval, the model then computes a mean estimate and a confidence interval for the average waiting time of customers between 0830 and 0845.

The model also computes the average number of cashiers on duty during each 15 minute interval. A confidence interval was not computed for this performance measure since the upper bound on the number of cashiers is already determined by the system. For example, if a store only has 20 checkout stands, the calculation of a 95% confidence interval might show that the number of cashiers on duty during a time interval will fall between 18 and 22, which is clearly not possible. Therefore it was decided to provide as output the average number of cashiers on duty for each 15 minute interval along with the minimum and maximum number of cashiers on duty during that interval over all runs of the simulation.

Sample Output

The sample output in Tables VIII, IX, and X was obtained by running the simulation model with the data collected at the Wright-Patterson AFB Commissary on October 23, 1987. A complete example of how to execute the model and obtain output is given in the GPCSS User's Manual in Appendix B. Interpretation of the output is covered next.

Table VIII. Part One of Sample Output

Run 1 of 5

CURRENT TIME 2015

FILE STATISTICS

FILE NO.	NODE LABEL	AVERAGE LENGTH	AVERAGE WAIT TIME	STANDARD DEVIATION	MAX LENGTH
1	PKNG	.173	.047	.923	9.0
2	DESK	.253	.068	1.655	37.0
3	CHEK	.000	.000	.000	.0
4	REG	9.020	2.681	14.302	56.0
5	XPRS	.000	.000	.000	1.0
6	XLIN	.325	.925	.710	5.0
7	LIN 1	47.523	28.465	24.321	97.0
8	LIN 2	47.078	27.774	24.212	97.0

STATISTICS FOR VARIABLES BASED ON OBSERVATION

VARIABLE NAME	MEAN VALUE	STD DEV	MIN VALUE	MAX VALUE	NO.OF OBS
PARK BALKS	.00	.00	.0	.0	.0
EXPRS TIS	29.77	13.21	6.7	67.3	248.0
LINE 1 TIS	96.80	28.34	13.0	215.8	1177.0
LINE 2 TIS	95.57	27.66	19.9	196.5	1195.0

RESOURCE STATISTICS

RESOURCE LABEL	RESOURCE UTIL	AVERAGE AVAIL	STANDARD DEV	MAXIMUM UTIL
PARK	333.91	116.09	137.21	450.00
SMAL	8.09	141.91	4.92	20.00
LRGE	357.26	92.74	149.40	450.00
XCSH	.52	.53	.53	1.00
CASH 1	8.02	.87	3.27	10.00
CASH 2	7.90	1.00	3.54	10.00
BAG 1	20.55	6.13	8.57	30.00
BAG 2	20.52	6.16	9.11	30.00

Table VIII. Part One of Sample Output (Continued)

ACTIVITY STATISTICS

ACTIVITY INDEX	AVERAGE UTIL	MAXIMUM UTIL	STANDARD DEVIATION	ENTITY COUNT
1	.000	1.000	.000	2582
2	7.325	18.000	4.428	2582
3	.372	4.400	.483	2620
4	.000	1.000	.000	1838
5	.000	1.000	.000	782
6	1.546	6.000	1.403	1838
7	.035	2.000	.185	248
8	.336	3.000	.533	2372
9	7.284	17.000	4.485	248
10	169.800	253.000	77.592	2372
11	.481	2.000	.522	248
12	1.407	5.000	1.114	248
13	7.855	10.000	3.230	1177
14	12.529	28.000	5.728	1177
15	7.726	10.000	3.500	1195
16	12.628	28.000	5.955	1195

Output Interpretation.The output contains all of the information found in a SLAM II Summary Report (22:145,267). For the sample output, the model was run five times. A summary report is given for each individual run. (Only one of the five summary reports is given here because of space considerations). In Table VIII, the first line in the report tells which run of the total number of runs the report represents. The next line shows in military time, when the simulation terminated. Following all of the individual summary reports are two other tables. The first, shown in Table IX, gives the mean estimates of customer waiting times with 95% confidence intervals. The second, shown in Table X, gives the average, minimum, and maximum number of cashiers on duty during each 15 minute interval. It was seen from the

Table IX. Part Two of Sample Output

Average Waiting Time In Queue Line

Mean Estimate	95% Confidence Interval Lower Limit	Upper Limit	Time
.00	.00	.00	845
.83	.28	1.37	900
4.97	2.90	7.04	915
11.37	9.10	13.64	930
16.82	13.88	19.77	945
20.59	17.83	23.34	1000
26.44	24.86	28.02	1015
30.95	28.01	33.88	1030
32.56	28.05	37.06	1045
36.19	29.44	42.94	1100
35.13	30.59	39.66	1115
35.50	27.24	43.77	1130
34.45	23.86	45.05	1145
33.54	24.05	43.02	1200
34.88	27.53	42.23	1215
32.56	22.59	42.53	1230
30.66	26.82	34.50	1245
32.24	26.88	37.61	1300
28.96	23.56	34.36	1315
27.35	19.43	35.27	1330
25.65	19.45	31.84	1345
29.23	25.03	33.43	1400
29.02	19.10	38.93	1415
27.18	14.32	40.05	1430
27.40	18.90	35.90	1445
30.76	23.44	38.07	1500
27.75	19.72	35.79	1515
29.74	22.61	36.88	1530
29.63	21.55	37.70	1545
30.19	24.42	35.96	1600
31.46	26.86	36.06	1615
33.19	28.89	37.49	1630
32.05	27.12	36.99	1645
24.74	20.14	35.34	1700
30.89	24.88	36.90	1715
26.65	14.28	39.03	1730
24.95	13.46	36.43	1745
22.84	16.64	29.05	1800

Table X. Part Three of Sample Output

Average Number of Cashiers on Duty

Mean Estimate	Low Value	High Value	Time
4.00	4.00	4.00	845
4.00	4.00	4.00	900
5.60	4.00	6.00	915
7.60	6.00	8.00	930
9.60	8.00	10.00	945
11.60	10.00	12.00	1000
13.60	12.00	14.00	1015
15.60	14.00	16.00	1030
17.60	16.00	18.00	1045
19.60	18.00	20.00	1100
20.00	20.00	20.00	1115
20.00	20.00	20.00	1130
20.00	20.00	20.00	1145
20.00	20.00	20.00	1200
20.00	20.00	20.00	1215
20.00	20.00	20.00	1230
20.00	20.00	20.00	1245
20.00	20.00	20.00	1300
20.00	20.00	20.00	1315
20.00	20.00	20.00	1330
20.00	20.00	20.00	1345
20.00	20.00	20.00	1400
20.00	20.00	20.00	1415
20.00	20.00	20.00	1430
20.00	20.00	20.00	1445
20.00	20.00	20.00	1500
20.00	20.00	20.00	1515
20.00	20.00	20.00	1530
20.00	20.00	20.00	1545
20.00	20.00	20.00	1600
20.00	20.00	20.00	1615
20.00	20.00	20.00	1630
20.00	20.00	20.00	1645
20.00	20.00	20.00	1700
20.00	20.00	20.00	1715
20.00	20.00	20.00	1730
20.00	20.00	20.00	1745
20.00	20.00	20.00	1800

output that the five runs did not terminate at the same time. This variability made it impossible to calculate average waiting times and confidence intervals past the point

in the simulation where the user had specified the commissary to close. For example, the output in Table IX had an opening time of 0830 and a closing time of 1800. The model can compute the average waiting times and confidence intervals for each 15 minute interval between these operating hours. The problem arises in computing these estimates after closing time since some runs continue longer than others. It was decided to calculate the mean estimates and confidence intervals for all 15 minute intervals during open hours and to calculate an overall average waiting time along with a 95% confidence interval after closing time. The overall waiting time of customers after closing is given directly after the table of estimates for each 15 minute interval.

The above approach seems reasonable because since after closing time no more customers are arriving to the store. Also, during data collection it was learned that when the store closes, the cashiers on duty stay at their registers until all customers have been serviced. The model thus assumes that the number of cashiers on duty after closing time does not change and the output report for number of cashiers on duty only covers the hours between opening time and closing time.

Summary

This chapter has covered model verification and validation. Definitions for these tasks were given and how each was accomplished in this research was discussed.

Finally, output analysis was covered, and a sample of model output was presented.

The next chapter concludes the report by presenting results, discussing conclusions, and making recommendations for further studies and model embellishments.

VI. Results, Recommendations, and Conclusions

Overview

This chapter concludes the report by discussing the results and conclusions of the research and by presenting recommendations for further studies and model improvements.

Results

The primary objective of this research effort was to develop a general purpose commissary store simulation model that would help commissary management more effectively manage commissary stores. The model developed was verified and validated and is a reasonable representation of a commissary system. The model produces output for the performance measures that were the main interest of commissary officials --the average waiting times of customers in checkout lines and the number of cashiers needed to keep customer waiting times in checkout lines below 15 minutes.

A subsidiary objective was to make the model easy to use. This was accomplished by making the model menu-driven, that is, the user executes the model by answering screen prompts and inputting data. A user's guide for the model was also developed and is given in Appendix B.

The remaining subsidiary objectives dealt with data collection and analysis. During data collection at the Wright-Patterson AFB Commissary, guidelines were developed to help future users of the model collect appropriate data. Statistical models were also developed that help predict how

many items customers will buy and how long it will take customers to shop and checkout.

In summary, the objectives of the research were met and the final model is a product that will help AFCOMS officials manage Air Force commissaries.

Recommendations

The development of a simulation model is an iterative process because new elements and more detail can be added to a model in successive versions of the model. The iterative approach was used in the development of the general purpose commissary store simulation model. Although the objectives of this research were met, there are other elements and details that AFCOMS should consider adding to the model. These additions are presented next in the following order : 1) incorporate logic in the program to calculate the number of additional runs needed to reach a specified accuracy; 2) incorporate in the model the capability to allow the user to set an upper bound on the number of cashiers available during each hour of the day; 3) conduct a more detailed study of congestion; and 4) conduct a detailed study on how time of day affects certain elements such as the number of items customers buy, the amount of time customers spend shopping, and cashier speed.

Automated Calculation of Additional Runs. The author tried to incorporate this item in the model in the last two weeks of the research period. Unfortunately, the time left to

finish this report was insufficient to allow the author to make this change. It is believed that with further investigation this worthwhile addition can be implemented in the model. The major benefit of adding this feature to the model would be that managers at local stores would be able to use the model without having to be knowledgeable about determining the number of runs needed to achieve a specified accuracy. For example, if a manager wanted to be 90% confident that enough cashiers had been scheduled to ensure customers would not have to wait longer than 15 minutes in line given an anticipated workload, the model would automatically calculate the number of runs needed to achieve the specified accuracy.

Setting Upper Bounds on Cashiers Available. The present model is dynamic and shows the user how many cashiers are needed to handle the workload and to meet the objective that no customer waits in line more than 15 minutes. The capability to set hourly upper bounds on the number of cashiers available for duty would give the user the ability to see how certain scheduling strategies would affect customer waiting time in the checkout queues.

To incorporate this feature into the model, the FORTRAN code could be changed in Subroutine Arvl so that the program would prompt the user to enter an hourly upper bound on the number of cashiers available at the same time customer arrival rates are entered.

Further Congestion Study. A scientific approach should be taken to study the effects of congestion in commissaries. Work has been done in the transportation field and may be applicable to stores (1). Also, as mentioned in Chapter IV, congestion could possibly be included as another variable in the regression equation that predicts shopping time.

Effects of Time of Day. This area would involve studies that investigate how the time of day affects the number of items customers buy, how long customers spend shopping, and the speed at which cashiers check customers out. Intuitively, it seems reasonable that customers who buy large amounts of groceries will shop in the early morning or mid afternoon. It is also likely that shoppers will spend more time shopping if the store is not crowded and they do not feel rushed. On the other hand, customers who shop late in the day or near closing time will probably shop faster than normal. In fact during data collection at the Wright-Patterson AFB store, the author observed that customers seemed to shop faster if they arrived at the store close to closing time.

Cashier speed will probably be affected by the time of day also. For example, if a cashier has been working for several hours, the speed at which they check customers out may deteriorate. Also, after closing time, the customers left in the store must still be checked out. It seems reasonable to expect that cashier speed will increase so that the remaining customers can be serviced and the work day

will end for the cashiers. All of these effects that the time of day may have could be studied through designing appropriate experiments and conducting analysis of variance.

Conclusions

The General Purpose Commissary Store Simulation Model can be a valuable tool for decision makers in the Air Force Commissary Service. The model can be used by management at AFCOMS to investigate the effect of different policies and operational considerations without actually implementing them. Examples of such investigations would include : 1) planning new stores; 2) sizing of checkout capabilities; 3) setting operating hours; and 4) scheduling of workers.

Since the model was developed in an iterative manner as discussed above, analysts at AFCOMS should be able to easily add new features and details to the model as deemed appropriate and necessary. This first version of the model is a valid and reasonable representation of a commissary system and will grow into an extremely useful management tool if used and refined by the expert managers and analysts at AFCOMS.

Appendix A: SLAM II and FORTRAN Code Listings

```
GEN,MOULDER,COMMISSARY,07/01/87,10,,N,,N,N,72;
LIMITS,41,3,1000;
INTLC,XX(2)=.7; PERCENT CHECK CUSTOMERS
INTLC,XX(3)=.6; TIME TO STAMP A CHECK
INTLC,XX(4)=.1; PERCENT EXPRESS CUSTOMERS
INTLC,XX(5)=5.51; MEAN CHECKOUT TIME
INTLC,XX(7)=300.; LOW CONGESTION THRESHOLD
INTLS,XX(9)=1.0; INITIAL CONGESTION FACTOR
INTLC,XX(10)=350; MEDIUM CONGESTION THRESHOLD
INTLC,XX(11)=400; HIGH CONGESTION THRESHOLD
INTLC,XX(17)=13.8; Y INT OF REGULAR SHOPPING TIME EQUATION
INTLC,XX(18)=.4; SLOPE OF REGULAR SHOPPING TIME EQUATION
INTLC,XX(19)=1.46; Y INT OF REGULAR CHECKOUT TIME EQUATION
INTLC,XX(20)=.05; SLOPE OF REGULAR CHECKOUT TIME EQUATION
INTLC,XX(21)=.81; Y INT OF EXPRESS CHECKOUT TIME EQUATION
INTLC,XX(22)=.04; SLOPE OF EXPRESS CHECKOUT TIME EQUATION
INTLC,XX(23)=11.38; MSE FOR REGULAR SHOPPING TIME
INTLC,XX(24)=1.8; MSE FOR REGULAR CHECKOUT TIME
INTLC,XX(25)=.46; MSE FOR EXPRESS CHECKOUT TIME
INTLC,XX(32)=10.0; TIME INTERVAL TO CHECK STATE OF SYSTEM
SEVNT,7,XX(6),XP,XX(7);
SEVNT,8,XX(6),XN,XX(7);
SEVNT,9,XX(6),XP,XX(10);
SEVNT,7,XX(6),XN,XX(10);
SEVNT,10,XX(6),XP,XX(11);
SEVNT,9,XX(6),XN,XX(11);
NETWORK;
;
;      PARKING SPACES, SHOPPING CARTS, BAGGERS, AND CASHIERS
;      ARE ALL TREATED AS RESOURCES. THE USER WILL ENTER THE
;      NUMBER OF EACH OF THESE RESOURCES IN THE FORTRAN
;      SUBROUTINE I N T L C. THE SLAM SUBROUTINES ALTER, SEIZE,
;      AND FILEM WILL BE USED TO CHANGE THE RESOURCES AS
;      NEEDED.
;
       RESOURCE/1,PARK(0),1;  PARKING LOT
       RESOURCE/2,SMAL(0),5; SMALL SHOPPING CARTS
       RESOURCE/3,LRGE(0),4; LARGE SHOPPIING CARTS
       RESOURCE/4,XCASH(1),6; CASHIERS FOR EXPRESS LINES
       RESOURCE/5,CASH1(0),7; CASHIERS FOR QUEUE LINE 1
       RESOURCE/6,CASH2(0),8; CASHIERS FOR QUEUE LINE 2
       RESOURCE/7,CASH3(0),9; CASHIERS FOR QUEUE LINE 3
       RESOURCE/8,CASH4(0),10; CASHIERS FOR QUEUE LINE 4
       RESOURCE/9,CASH5(0),11; CASHIERS FOR QUEUE LINE 5
       RESOURCE/10,CASH6(0),12; CASHIERS FOR QUEUE LINE 6
       RESOURCE/11,CASH7(0),13; CASHIERS FOR QUEUE LINE 7
       RESOURCE/12,CASH8(0),14; CASHIERS FOR QUEUE LINE 8
       RESOURCE/13,CASH9(0),15; CASHIERS FOR QUEUE LINE 9
       RESOURCE/14,CASH10(0),16; CASHIERS FOR QUEUE LINE 10
```

```
RESOURCE/15,CASH11(0),17; CASHIERS FOR QUEUE LINE 11
RESOURCE/16,CASH12(0),18; CASHIERS FOR QUEUE LINE 12
RESOURCE/17,CASH13(0),19; CASHIERS FOR QUEUE LINE 13
RESOURCE/18,CASH14(0),20; CASHIERS FOR QUEUE LINE 14
RESOURCE/19,CASH15(0),21; CASHIERS FOR QUEUE LINE 15
RESOURCE/20,CASH16(0),22; CASHIERS FOR QUEUE LINE 16
RESOURCE/21,CASH17(0),23; CASHIERS FOR QUEUE LINE 17
RESOURCE/22,CASH18(0),24; CASHIERS FOR QUEUE LINE 18
RESOURCE/23,CASH19(0),25; CASHIERS FOR QUEUE LINE 19
RESOURCE/24,CASH20(0),26; CASHIERS FOR QUEUE LINE 20
RESOURCE/25,CASH21(0),27; CASHIERS FOR QUEUE LINE 21
RESOURCE/26,CASH22(0),28; CASHIERS FOR QUEUE LINE 22
RESOURCE/27,CASH23(0),29; CASHIERS FOR QUEUE LINE 23
RESOURCE/28,CASH24(0),30; CASHIERS FOR QUEUE LINE 24
RESOURCE/29,CASH25(0),31; CASHIERS FOR QUEUE LINE 25
RESOURCE/30,CASH26(0),32; CASHIERS FOR QUEUE LINE 26
RESOURCE/31,CASH27(0),33; CASHIERS FOR QUEUE LINE 27
RESOURCE/32,CASH28(0),34; CASHIERS FOR QUEUE LINE 28
RESOURCE/33,CASH29(0),35; CASHIERS FOR QUEUE LINE 29
RESOURCE/34,CASH30(0),36; CASHIERS FOR QUEUE LINE 30
RESOURCE/35,CASH31(0),37; CASHIERS FOR QUEUE LINE 31
RESOURCE/36,CASH32(0),38; CASHIERS FOR QUEUE LINE 32
RESOURCE/37,CASH33(0),39; CASHIERS FOR QUEUE LINE 33
RESOURCE/38,CASH34(0),40; CASHIERS FOR QUEUE LINE 34
RESOURCE/39,CASH35(0),41; CASHIERS FOR QUEUE LINE 35
RESOURCE/40,BAG1(0),7; BAGGERS FOR QUEUE LINE 1
RESOURCE/41,BAG2(0),8; BAGGERS FOR QUEUE LINE 2
RESOURCE/42,BAG3(0),9; BAGGERS FOR QUEUE LINE 3
RESOURCE/43,BAG4(0),10; BAGGERS FOR QUEUE LINE 4
RESOURCE/44,BAG5(0),11; BAGGERS FOR QUEUE LINE 5
RESOURCE/45,BAG6(0),12; BAGGERS FOR QUEUE LINE 6
RESOURCE/46,BAG7(0),13; BAGGERS FOR QUEUE LINE 7
RESOURCE/47,BAG8(0),14; BAGGERS FOR QUEUE LINE 8
RESOURCE/48,BAG9(0),15; BAGGERS FOR QUEUE LINE 9
RESOURCE/49,BAG10(0),16; BAGGERS FOR QUEUE LINE 10
RESOURCE/50,BAG11(0),17; BAGGERS FOR QUEUE LINE 11
RESOURCE/51,BAG12(0),18; BAGGERS FOR QUEUE LINE 12
RESOURCE/52,BAG13(0),19; BAGGERS FOR QUEUE LINE 13
RESOURCE/53,BAG14(0),20; BAGGERS FOR QUEUE LINE 14
RESOURCE/54,BAG15(0),21; BAGGERS FOR QUEUE LINE 15
RESOURCE/55,BAG16(0),22; BAGGERS FOR QUEUE LINE 16
RESOURCE/56,BAG17(0),23; BAGGERS FOR QUEUE LINE 17
RESOURCE/57,BAG18(0),24; BAGGERS FOR QUEUE LINE 18
RESOURCE/58,BAG19(0),25; BAGGERS FOR QUEUE LINE 19
RESOURCE/59,BAG20(0),26; BAGGERS FOR QUEUE LINE 20
RESOURCE/60,BAG21(0),27; BAGGERS FOR QUEUE LINE 21
RESOURCE/61,BAG22(0),28; BAGGERS FOR QUEUE LINE 22
RESOURCE/62,BAG23(0),29; BAGGERS FOR QUEUE LINE 23
RESOURCE/63,BAG24(0),30; BAGGERS FOR QUEUE LINE 24
RESOURCE/64,BAG25(0),31; BAGGERS FOR QUEUE LINE 25
RESOURCE/65,BAG26(0),32; BAGGERS FOR QUEUE LINE 26
RESOURCE/66,BAG27(0),33; BAGGERS FOR QUEUE LINE 27
RESOURCE/67,BAG28(0),34; BAGGERS FOR QUEUE LINE 28
```

```
      RESOURCE/68,BAG29(0),35; BAGGERS FOR QUEUE LINE 29
      RESOURCE/69,BAG30(0),36; BAGGERS FOR QUEUE LINE 30
      RESOURCE/70,BAG31(0),37; BAGGERS FOR QUEUE LINE 31
      RESOURCE/71,BAG32(0),38; BAGGERS FOR QUEUE LINE 32
      RESOURCE/72,BAG33(0),39; BAGGERS FOR QUEUE LINE 33
      RESOURCE/73,BAG34(0),40; BAGGERS FOR QUEUE LINE 34
      RESOURCE/74,BAG35(0),41; BAGGERS FOR QUEUE LINE 35
;
;
;     THE FOLLOWING CREATE NODE PROVIDES CUSTOMER ARRIVALS.
;     GLOBAL VARIABLE XX(1) IS EQUAL TO THE NUMBER OF
;     MINUTES THE STORE WILL BE OPEN. XX(1) IS COMPUTED
;     IN THE FORTRAN SUBROUTINE I N T L C.
;     THE CONDITION THAT TNOW BE LESS THAN OR EQUAL TO
;     CLOSING TIME PROVIDES FOR A CUT-OFF OF CUSTOMER
;     ARRIVALS AFTER CLOSING TIME. THIS SIMULATES THE
;     CLOSING OF THE DOOR. SINCE NO INITIALIZE STATEMENT
;     IS USED IN THE NETWORK, THE SIMULATION WILL TERMINATE
;     AFTER ALL EVENTS ON THE EVENT CALENDER HAVE BEEN
;     COMPLETED, THAT IS, ALL CUSTOMERS HAVE FINISHED
;     SHOPPING AND HAVE LEFT THE PARKING LOT.
;
STRT  CREATE,,.1,1,,1;
      ACT,,TNOW.LE.XX(1),DOOR;
      ACT,,TNOW.GT.XX(1);
      EVENT,3,1;
      TERM;
DOOR  GOON;
      ACT,USERF(1),,STRT;
      ACT/1;
PKNG  AWAIT(1/50),PARK/1,BALK(LEV1); WAIT FOR PARKING
      ACT/2,2,,DESK;
LEV1  GOON; BALK AND LEAVE IF PARKING LOT TOO FULL
      COLCT(1),INT(1),PARK BALKS,,1;
      TERM;
DESK  QUEUE(2),0,99,BALK(LEV2); I.D. DESK
      ACT/3,.1,,CONT;
LEV2  GOON; BALK AND LEAVE IF LINE AT DESK TOO LONG
      FREE,PARK/1,1;
      TERM;
CONT  GOON,1;
      ACT/4,,XX(2),CHEK; CHECK WRITING CUSTOMERS
      ACT/5,,1-XX(2),CART; CASH CUSTOMERS
CHEK  GOON,1;
      ACT/6,XX(3);
CART  GOON,1;
      ACT/7,.1,XX(4),XPRS; EXPRESS CUSTOMERS
      ACT/8,.1,1-XX(4),REG; REGULAR CUSTOMERS
XPRS  EVENT,12,1;
      AWAIT(5),SMAL/1,BLOCK,1;
      ACT/9,USERF(2),,XLIN;
REG   EVENT,11,1;
      AWAIT(4),LRGE/ATRIB(3),BLOCK,1;
```

```
       ACT/10,USERF(2),,SELQ;
;
;    THE FOLLOWING PART OF THE NETWORK SIMULATES THE EXPRESS
; LINE CHECKOUT. IT IS ASSUMED THERE IS ALWAYS ONE CASHIER AND ONE
; BAGGER AT THE EXPRESS LINE. AFTER THE CUSTOMER IS SERVICED, THEY
; LEAVE THE STORE AND THE PARKING LOT, FREEING A SMALL CART AND
; A PARKING SPOT.
;
XLIN   AWAIT(6),XCASH;
       ACT/11,USERF(4);
       FREE,SMAL/1,1;
       ACT,.1;
       FREE,XCASH/1,1;
       ACT/12,4;
       FREE,PARK/1,1;
       COLCT(6),INT(1),EXPRESS TIS,,1;
       TERM;
;
;   THE FOLLOWING PART OF THE NETWORK SIMULATES THE REGULAR QUEUE
; LINES. IT IS ASSUMED THAT CUSTOMERS WILL CHOOSE THE SHORTEST QUEUE
; LINE AND THEREFORE THE NUMBER OF CUSTOMERS EXPECTED TO JOIN A
; QUEUE WILL BE (1/N)*TOTAL # CUSTOMERS, WHERE N IS THE NUMBER OF
; QUEUEING LINES IN THE STORE. PUT ANOTHER WAY, THE PERCENTAGE OF
; CUSTOMERS EXPECTED TO JOIN A QUEUE LINE IS 1/N, WHERE N IS THE
; NUMBER OF QUEUEING LINES IN THE STORE.
;
SELQ   GOON,1;
       EVENT,1,1;
       TERM:
LIN1   AWAIT(7),ALLOC(1); LINE 1 QUEUE
       ACT/13,USERF(3); CHECKOUT TIME
       FREE,LRGE/ATRIB(3),1;
       ACT,.1;
       FREE,CASH1/1,1;
       ACT/14,UNFRM(5,15); TRANSIT TIME FOR BAGGER
       FREE,BAG1/1,1;
       ACT;
       FREE,PARK/1,1;
       COLCT(7),INT(1),LINE 1 TIS,,1;
       TERM;
LIN2   AWAIT(8),ALLOC(2);LINE 2 QUEUE
       ACT/15,USERF(3); CHECKOUT TIME
       FREE,LRGE/ATRIB(3),1;
       ACT,.1;
       FREE,CASH2/1,1;
       ACT/16,UNFRM(5,15); TRANSIT TIME FOR BAGGER
       FREE,BAG2/1,1;
       ACT;
       FREE,PARK/1,1;
       COLCT(8),INT(1),LINE 2 TIS,,1;
       TERM;
LIN3   AWAIT(9),ALLOC(3);LINE 3 QUEUE
       ACT/17,USERF(3);
```

```
       FREE,LRGE/ATRIB(3),1;
       ACT,.1;
       FREE,CASH3/1,1;
       ACT/18,UNFRM(5,15); TRANSIT TIME FOR BAGGER
       FREE,BAG3/1,1;
       ACT;
       FREE,PARK/1,1;
       COLCT(9),INT(1),LINE 3 TIS,,1;
       TERM;
LIN4   AWAIT(10),ALLOC(4); LINE 4 QUEUE
       ACT/19,USERF(3);
       FREE,LRGE/ATRIB(3),1;
       ACT,.1;
       FREE,CASH4/1,1;
       ACT/20,UNFRM(5,15); TRANSIT TIME FOR BAGGER
       FREE,BAG4/1,1;
       ACT;
       FREE,PARK/1,1;
       COLCT(10),INT(1),LINE 4 TIS,,1;
       TERM;
LIN5   AWAIT(11),ALLOC(5); LINE 1 QUEUE
       ACT/21,USERF(3);
       FREE,LRGE/ATRIB(3),1;
       ACT,.1;
       FREE,CASH5/1,1;
       ACT/22,UNFRM(5,15); TRANSIT TIME FOR BAGGER
       FREE,BAG5/1,1;
       ACT;
       FREE,PARK/1,1;
       COLCT(11),INT(1),LINE 5 TIS,,1;
       TERM;
LIN6   AWAIT(12),ALLOC(6); LINE 6 QUEUE
       ACT/23,USERF(3);
       FREE,LRGE/ATRIB(3),1;
       ACT,.1;
       FREE,CASH6/1,1;
       ACT/24,UNFRM(5,15); TRANSIT TIME FOR BAGGER
       FREE,BAG6/1,1;
       ACT;
       FREE,PARK/1,1;
       COLCT(12),INT(1),LINE 6 TIS,,1;
       TERM;
LIN7   AWAIT(13),ALLOC(7); LINE 7 QUEUE
       ACT/25,USERF(3);
       FREE,LRGE/ATRIB(3),1;
       ACT,.1;
       FREE,CASH7/1,1;
       ACT/26,UNFRM(5,15); TRANSIT TIME FOR BAGGER
       FREE,BAG7/1,1;
       ACT;
       FREE,PARK/1,1;
       COLCT(13),INT(1),LINE 7 TIS,,1;
       TERM;
```

```
LIN8  AWAIT(14),ALLOC(8); LINE 8 QUEUE
      ACT/27,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH8/1,1;
      ACT/28,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG8/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(14),INT(1),LINE 8 TIS,,1;
      TERM;
LIN9  AWAIT(15),ALLOC(9); LINE 9 QUEUE
      ACT/29,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH9/1,1;
      ACT/30,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG9/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(15),INT(1),LINE 9 TIS,,1;
      TERM;
LI10  AWAIT(16),ALLOC(10); LINE 10 QUEUE
      ACT/31,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH10/1,1;
      ACT/32,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG10/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(16),INT(1),LINE 10 TIS,,1;
      TERM;
LI11  AWAIT(17),ALLOC(11); LINE 11 QUEUE
      ACT/33,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH11/1,1;
      ACT/34,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG11/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(17),INT(1),LINE 11 TIS,,1;
      TERM;
LI12  AWAIT(18),ALLOC(12); LINE 12 QUEUE
      ACT/35,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH12/1,1;
      ACT/36,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG12/1,1;
      ACT;
      FREE,PARK/1,1;
```

```
      COLCT(18),INT(1),LINE 12 TIS,,1;
      TERM;
LI13  AWAIT(19),ALLOC(13); LINE 13 QUEUE
      ACT/37,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH13/1,1;
      ACT/38,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG13/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(19),INT(1),LINE 13 TIS,,1;
      TERM;
LI14  AWAIT(20),ALLOC(14); LINE 14 QUEUE
      ACT/39,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH14/1,1;
      ACT/40,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG14/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(20),INT(1),LINE 14 TIS,,1;
      TERM;
LI15  AWAIT(21),ALLOC(15); LINE 15 QUEUE
      ACT/41,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH15/1,1;
      ACT/42,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG15/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(21),INT(1),LINE 15 TIS,,1;
      TERM;
LI16  AWAIT(22),ALLOC(16); LINE 16 QUEUE
      ACT/43,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH16/1,1;
      ACT/44,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG16/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(22),INT(1),LINE 16 TIS,,1;
      TERM;
LI17  AWAIT(23),ALLOC(17); LINE 17 QUEUE
      ACT/45,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH17/1,1;
      ACT/46,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG17/1,1;
```

```
      ACT;
      FREE,PARK/1,1;
      COLCT(23),INT(1),LINE 17 TIS,,1;
      TERM;
LI18  AWAIT(24),ALLOC(18); LINE 18 QUEUE
      ACT/47,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH18/1,1;
      ACT/48,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG18/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(24),INT(1),LINE 18 TIS,,1;
      TERM;
LI19  AWAIT(25),ALLOC(19); LINE 19 QUEUE
      ACT/49,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH19/1,1;
      ACT/50,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG19/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(25),INT(1),LINE 19 TIS,,1;
      TERM;
LI20  AWAIT(26),ALLOC(20); LINE 20 QUEUE
      ACT/51,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH20/1,1;
      ACT/52,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG20/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(26),INT(1),LINE 20 TIS,,1;
      TERM;
LI21  AWAIT(27),ALLOC(21); LINE 21 QUEUE
      ACT/53,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH21/1,1;
      ACT/54,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG21/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(27),INT(1),LINE 21 TIS,,1;
      TERM;
LI22  AWAIT(28),ALLOC(22); LINE 22 QUEUE
      ACT/55,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH22/1,1;
```

```
      ACT/56,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG22/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(28),INT(1),LINE 22 TIS,,1;
      TERM;
LI23  AWAIT(29),ALLOC(23); LINE 23 QUEUE
      ACT/57,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH23/1,1;
      ACT/58,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG23/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(29),INT(1),LINE 23 TIS,,1;
      TERM;
LI24  AWAIT(30),ALLOC(24); LINE 24 QUEUE
      ACT/59,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH24/1,1;
      ACT/60,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG24/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(30),INT(1),LINE 24 TIS,,1;
      TERM;
LI25  AWAIT(31),ALLOC(25); LINE 25 QUEUE
      ACT/61,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH25/1,1;
      ACT/62,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG25/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(31),INT(1),LINE 25 TIS,,1;
      TERM;
LI26  AWAIT(32),ALLOC(26); LINE 26 QUEUE
      ACT/63,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH26/1,1;
      ACT/64,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG26/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(32),INT(1),LINE 26 TIS,,1;
      TERM;
LI27  AWAIT(33),ALLOC(27); LINE 27 QUEUE
      ACT/65,USERF(3);
      FREE,LRGE/ATRIB(3),1;
```

```
      ACT,.1;
      FREE,CASH27/1,1;
      ACT/66,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG27/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(33),INT(1),LINE 27 TIS,,1;
      TERM;
LI28  AWAIT(34),ALLOC(28); LINE 28 QUEUE
      ACT/67,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH28/1,1;
      ACT/68,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG28/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(34),INT(1),LINE 28 TIS,,1;
      TERM;
LI29  AWAIT(35),ALLOC(29); LINE 29 QUEUE
      ACT/69,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH29/1,1;
      ACT/70,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG29/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(35),INT(1),LINE 29 TIS,,1;
      TERM;
LI30  AWAIT(36),ALLOC(30); LINE 30 QUEUE
      ACT/71,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH30/1,1;
      ACT/72,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG30/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(36),INT(1),LINE 30 TIS,,1;
      TERM;
LI31  AWAIT(37),ALLOC(31); LINE 31 QUEUE
      ACT/73,USERF(3);
      FREE,LRGE/ATRIB(3),1;
      ACT,.1;
      FREE,CASH31/1,1;
      ACT/74,UNFRM(5,15); TRANSIT TIME FOR BAGGER
      FREE,BAG31/1,1;
      ACT;
      FREE,PARK/1,1;
      COLCT(37),INT(1),LINE 31 TIS,,1;
      TERM;
LI32  AWAIT(38),ALLOC(32); LINE 32 QUEUE
```

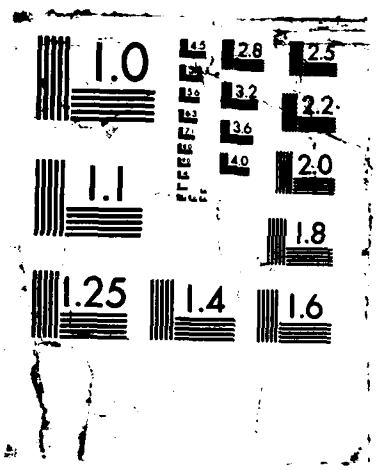
```
        ACT/75,USERF(3);
        FREE,LRGE/ATRIB(3),1;
        ACT,.1;
        FREE,CASH32/1,1;
        ACT/76,UNFRM(5,15); TRANSIT TIME FOR BAGGER
        FREE,BAG32/1,1;
        ACT;
        FREE,PARK/1,1;
        COLCT(38),INT(1),LINE 33 TIS,,1;
        TERM;
LI33    AWAIT(39),ALLOC(34); LINE 34 QUEUE
        ACT/77,USERF(3)
        FREE,LRGE/ATRIB(3),1;
        ACT,.1;
        FREE,CASH33/1,1;
        ACT/78,UNFRM(5,15); TRANSIT TIME FOR BAGGER
        FREE,BAG33/1,1;
        ACT;
        FREE,PARK/1,1;
        COLCT(39),INT(1),LINE 33 TIS,,1;
        TERM;
LI34    AWAIT(40),ALLOC(1); LINE 34 QUEUE
        ACT/79,USERF(3);
        FREE,LRGE/ATRIB(3),1;
        ACT,.1;
        FREE,CASH34/1,1;
        ACT/80,UNFRM(5,15); TRANSIT TIME FOR BAGGER
        FREE,BAG34/1,1;
        ACT;
        FREE,PARK/1,1;
        COLCT(40),INT(1),LINE 34 TIS,,1;
        TERM;
LI35    AWAIT(41),ALLOC(35); LINE 35 QUEUE
        ACT/81,USERF(3);
        FREE,LRGE/ATRIB(3),1;
        ACT,.1;
        FREE,CASH35/1,1;
        ACT/82,UNFRM(5,15); TRANSIT TIME FOR BAGGER
        FREE,BAG35/1,1;
        ACT;
        FREE,PARK/1,1;
        COLCT(41),INT(1),LINE 35 TIS,,1;
        TERM;
        END;
SEEDS,13711561(1),18490213(2),9765517(3),
        57892031(4),96221977(5),3799123(6),
        87593311(7),89721003(8);
SIMULATE;
SEEDS,78493371(1),45612997(2),90199755(3),
        34501255(4),59013277(5),9812377(6),
        91023711(7),87819917(8);
SIMULATE;
SEEDS,7329121(1),8713119(2),7263971(3),
```

```
       8109231(4),9553107(5),9734103(6),
       1845737(7),5487233(8);
SIMULATE;
SEEDS,4910537(1),9908345(2),8710331(3),
       2091445(4),9890127(5),6673377(6),
       3012354(7),6371031(8);
SIMULATE;
SEEDS,8603521(1),3499771(2),1819573(3),
       5120935(4),8924115(5),7702341(6),
       2001773(7),3289567(8);
SIMULATE;
SEEDS,348713(1),189377(2),456131(3),
       94501(4),781033(5),870139(6),
       62013(7),762291(8);
SIMULATE;
SEEDS,209771(1),904563(2),671135(3),
       891033(4),118677(5),892331(6),
       207737(7),189433(8);
SIMULATE;
SEEDS,548323(1),769127(2),084551(3),
       189743(4),901337(5),785437(6),
       386447(7),299811(8);
SIMULATE;
SEEDS,789123(1),442713(2),431173(3),
       873291(4),665137(5),149233(6),
       634881(7),736641(8);
SIMULATE;
SEEDS,981743(1),784621(2),378427(3),
       872653(4),985515(5),426355(6),
       980137(7),871123(8);
SIMULATE;
FIN;
```

AD-A189 502 DEVELOPMENT OF A GENERAL PURPOSE COMMISSARY STORE SIMULATION MODEL(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING R D MOULDER DEC 87 AFIT/GOR/ENS/87D-11 2/2

UNCLASSIFIED F/G 15/5 NL

END
DATE
FILMED
-8-


1.0
4.5
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

```
#INCLUDE:'PRCTL.FOR'
C
*********************************************************
C
C  This is the FORTRAN interface with the SLAM EXECUTIVE.
C  All of the subroutines are used for interfacing with
C  the user or for changing model resources during
C  execution of the Simulation.
C
C********************************************************
      PROGRAM MAIN
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      NCRDR=5
      NPRNT=0
      NTAPE=7
      CALL SLAM
      STOP ' '
      END
C
C
C
      SUBROUTINE ALLOC(I,IFLAG)
C
*********************************************************
C
C This Subroutine allocates resources, i.e. cashiers and
C baggers to serve customers that have finished shopping.
C
C********************************************************
C
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      IFLAG=0
C
C********************************************************
C
C   If a cashier and a bagger are not BOTH available then
C   service cannot begin, else if both are available seize
C   one of each and service begins.
C
C********************************************************
C
      IF(NNRSC(I+39).LE.0.OR.NNRSC(I+4).LE.0)RETURN
      CALL SEIZE(I+39,1)
      CALL SEIZE(I+4,1)
      IFLAG=-1
      RETURN
      END

         SUBROUTINE INTLC
**********************************************************
C
```

```
C    The purpose of this subroutine is to allow the user
C    to set up a configuration file so that a commissary
C    store can be simulated. The approach is generally one
C    of giving the user options to choose from so that they
C    can configure any store by entering values for the
C    appropriate parameters that identify that store. These
C    values are saved in a file so that if the user wants to
C    redo the same simulation later then the values will not
C    have to be re-entered.
C
C**********************************************************
C
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/NSTPSN/A(1000,2),RATEIN(1000,2),SSUBN,NPTS,TIME
      COMMON/USER1/NQUE,NOPT,NSTAND(38),NSTRT(38),NQUSE,C(38)
      COMMON/USER2/NOPEN,NCLOSE
      CHARACTER*12 FILENAM
      CHARACTER*12 FIL2
      CHARACTER*2 DUMMY
         ATRIB(1)=60.0
         CALL SCHDL(2,XX(32),ATRIB)
         CALL SCHDL(5,XX(32),ATRIB)
         ATRIB(1)=0.0
      CALL SCHDL(4,XX(32),ATRIB)
      XX(9)=1.0
      XX(30)=0.0
      IF(NNRUN.EQ.1)THEN
        OPEN(4,FILE='AVG.DAT',STATUS='NEW')
        OPEN(10,FILE='AVAFT.DAT',STATUS='NEW')
      ENDIF
      CALL BLANK
      WRITE(*,*)' '
C
***************************************************************
C
C    The user is prompted to enter their choice for the current
C    simulation.
C
***************************************************************
C
       IF(NNRUN.GT.1)THEN
                OPEN(2,FILE=FILENAM,STATUS='OLD')
                READ(2,*)NPARK,NLRGE,NSMAL,NBAG,NQUSE,NWAIT,NOPEN,NCLOSE
                READ(2,*)NQUE
                DO 200 I=1,NQUSE
                  READ(2,*)NSTAND(I)
200             CONTINUE
                DO 210 I=1,NQUE
                  READ(2,*)NSTRT(I)
210             CONTINUE
                IDIFF=NCLOSE-NOPEN
                IDIV=IDIFF/100
```

```
                 IMULT=IDIV*100
                 NEND=(NCLOSE-NOPEN)/100
                 IF(IDIFF.NE.IMULT)NEND=NEND+1
                 DO 220 I=1,NEND
                   READ(2,*)C(I)
220              CONTINUE
        XX(1)=(REAL((NCLOSE-NOPEN)/100)*100)*.6
        IDIFF=((NCLOSE-NOPEN)/100)*100
        DIFF=REAL(NCLOSE)-REAL(NOPEN)
        IF(REAL(IDIFF).NE.DIFF)THEN
          EXTRA=DIFF-REAL(IDIFF)
          IF(EXTRA.GE.60.0)EXTRA=EXTRA-40.0
          XX(1)=XX(1)+EXTRA
        ENDIF
                 GO TO 71
        ENDIF
5       WRITE(*,*)' THE FOLLOWING ARE YOUR OPTIONS FOR THIS SIMULATION:'
        WRITE(*,*)' '
        WRITE(*,*)'   1 - Build a NEW Configuration File'
        WRITE(*,*)'   2 - Run a Simulation using an EXISTING'
        WRITE(*,*)'       Configuration File'
        WRITE(*,*)'   3 - EXIT '
        WRITE(*,*)' '
        WRITE(*,'(A\)')' Enter the number of the desired option --> '
        READ(*,*)NOPT
        CALL BLANK
        IF(NOPT.EQ.1)THEN
          WRITE(*,'(A\)')' Enter the Name of the NEW File -->'
          READ(*,'(A12)')FILENAM
          OPEN(2,FILE=FILENAM,STATUS='NEW')
          GO TO 6
        ELSEIF(NOPT.EQ.2)THEN
          WRITE(*,'(A\)')' Enter the name of the EXISTING File -->'
          READ(*,'(A12)')FILENAM
          OPEN(2,FILE=FILENAM,STATUS='OLD')
          READ(2,*)NPARK,NLRGE,NSMAL,NBAG,NQUSE,NWAIT,
     1    NOPEN,NCLOSE
          WRITE(*,*)' '
          WRITE(*,*)' The Following will be used as inputs ...'
          GO TO 4
        ELSEIF(NOPT.EQ.3)THEN
          STOP
          RETURN
        ELSE
          CALL ERRER
          CALL BLANK
          GO TO 5
        ENDIF
6       CALL BLANK
C
C***********************************************************
C
C       The user is prompted to enter the parameter values
```

```
C      for a new configuration file. After entry, the values
C      the user entered are displayed and the opportunity
C      to change any of the values is offered.
C
C***********************************************************
C
       WRITE(*,*)' The following prompts will enable you to'
       WRITE(*,*)' configure a Commissary Store. The store will'
       WRITE(*,*)' be simulated using your inputs.'
       WRITE(*,*)' '
       WRITE(*,*)' '
       WRITE(*,'(A\)') '    Enter the number of Parking Spots --> '
       READ(*,*)NPARK
       WRITE(*,'(A\)') '    Enter the number of Large Carts  ---> '
       READ(*,*)NLRGE
       WRITE(*,'(A\)') '    Enter the number of Small Carts ----> '
       READ(*,*)NSMAL
       WRITE(*,'(A\)') '    Enter the number of Baggers --------> '
       READ(*,*)NBAG
       WRITE(*,'(A\)') '    Enter the number of Queue Lines ----> '
       READ(*,*)NQUSE
       WRITE(*,'(A\)') '    Number of waiting Customers --------> '
       READ(*,*)NWAIT
       WRITE(*,'(A\)') '    Enter Opening Time -----------------> '
       READ(*,*)NOPEN
       WRITE(*,'(A\)') '    Enter Closing Time -----------------> '
       READ(*,*)NCLOSE
       WRITE(2,*)NPARK,NLRGE,NSMAL,NBAG,NQUSE,NWAIT,
     1 NOPEN,NCLOSE
       CALL BLANK
       WRITE(*,*)' The following are your inputs ...'
4      WRITE(*,*)' '
       WRITE(*,*)' # Parking Spots      :',NPARK
       WRITE(*,*)' # Large Carts        :',NLRGE
       WRITE(*,*)' # Small Carts        :',NSMAL
       WRITE(*,*)' # Baggers            :',NBAG
       WRITE(*,*)' # Queue Lines        :',NQUSE
       WRITE(*,*)' # Waiting Customers :',NWAIT
       WRITE(*,*)' Opening Time         :',NOPEN
       WRITE(*,*)' Closing Time         :',NCLOSE
       WRITE(*,*)' '
       WRITE(*,'(A\)')' Are they correct ? [0-No, 1-Yes] --> '
       READ(*,*)NANSWR
          WRITE(*,*)' '
          WRITE(*,*)' '
          IF(NANSWR.EQ.1)GO TO 3
          IF(NANSWR.EQ.0)GO TO 2
                CALL ERRER
                CALL BLANK
                WRITE(*,*)' The following are your inputs ...'
             GO TO 4
2      WRITE(*,*)' Which of the following is incorrect ? :'
       WRITE(*,*)' '
```

```
      write(*,*)'      1 - Number of Parking Spots'
      WRITE(*,*)'      2 - Number of Large Carts'
      WRITE(*,*)'      3 - Number of Small Carts'
      WRITE(*,*)'      4 - Number of Baggers'
      WRITE(*,*)'      5 - Number of Queue Lines'
      WRITE(*,*)'      6 - Number of Waiting Customers'
      WRITE(*,*)'      7 - Opening Time'
      WRITE(*,*)'      8 - Closing Time'
      WRITE(*,*)' '
      WRITE(*,'(A\)')' Enter the Number of the incorrect value --> '
      READ(*,*)NWRONG
      IF(NWRONG.EQ.1)THEN
        WRITE(*,'(A\)')' Re-enter the number of Parking Spots --> '
        READ(*,*)NPARK
      ELSEIF(NWRONG.EQ.2)THEN
        WRITE(*,'(A\)')' Re-enter the number of Large Carts --> '
        READ(*,*)NLRGE
      ELSEIF(NWRONG.EQ.3)THEN
        WRITE(*,'(A\)')' Re-enter the number of Small Carts --> '
        READ(*,*)NSMAL
      ELSEIF(NWRONG.EQ.4)THEN
        WRITE(*,'(A\)')' Re-enter the number of Baggers --> '
        READ(*,*)NBAG
      ELSEIF(NWRONG.EQ.5)THEN
        IF(NOPT.EQ.2)THEN
            READ(2,*)NQUE
               DO 114 JJ=1,NQUSE
                 READ(2,*)NDUM
114         CONTINUE
            DO 116 JJ=1,NQUE
              READ(2,*)NDUM
116         CONTINUE
            NOPT2=5
         ENDIF
        WRITE(*,'(A\)')' Re-enter the number of Queue Lines --> '
        READ(*,*)NQUSE
      ELSEIF(NWRONG.EQ.6)THEN
        WRITE(*,'(A\)')' Re-enter number of Waiting Customers --> '
        READ(*,*)NWAIT
      ELSEIF(NWRONG.EQ.7)THEN
        WRITE(*,'(A\)')' Re-enter Opening Time --> '
        READ(*,*)NOPEN
        IF(NOPT.EQ.2)NOPT=4
      ELSEIF(NWRONG.EQ.8)THEN
        WRITE(*,'(A\)')' Re-enter Closing Time --> '
        READ(*,*)NCLOSE
        IF(NOPT.EQ.2)NOPT=4
      ELSE
        WRITE(*,*)' '
        WRITE(*,*)' '
        CALL ERRER
        CALL BLANK
        GO TO 2
```

```
      ENDIF
      CALL BLANK
      WRITE(*,*)' After correction, your inputs are now ...'
      GO TO 4
C
**********************************************************
C
C     At this point the user has entered the GENERAL
C     information about the Commissary to be simulated.
C     The user is now prompted to enter FRONT END information.
C
C*********************************************************
C
3     XX(1)=(REAL((NCLOSE-NOPEN)/100)*100)*.6
      IDIFF=((NCLOSE-NOPEN)/100)*100
      DIFF=REAL(NCLOSE)-REAL(NOPEN)
      IF(REAL(IDIFF).NE.DIFF)THEN
        EXTRA=DIFF-REAL(IDIFF)
        IF(EXTRA.GE.60.0)EXTRA=EXTRA-40.0
        XX(1)=XX(1)+EXTRA
      ENDIF
         CALL BLANK
         IF(NOPT.EQ.2.OR.NOPT.EQ.4)THEN
        IF(NOPT2.NE.5)THEN
             READ(2,*)NQUE
             DO 51 I=1,NQUSE
                READ(2,*)NSTAND(I)
51           CONTINUE
             DO 52 I=1,NQUE
                READ(2,*)NSTRT(I)
52           CONTINUE
           WRITE(*,*)' The following will be used as inputs ...'
           GO TO 40
          ENDIF
           ENDIF
       IF(NOPT2.EQ.5)THEN
            CALL BLANK
            WRITE(*,*)' Since you changed the number of Queue Lines'
            WRITE(*,*)' you must re-enter all FRONT END information'
            WRITE(*,*)' '
            WRITE(*,'(A\)')' Hit RETURN to continue --> '
            READ(*,'(A1)')DUMMY
            CALL BLANK
       ENDIF
         WRITE(*,*)' You will now be prompted to enter FRONT END'
         WRITE(*,*)' information, i.e. number of checkstands and'
         WRITE(*,*)' which Queue Lines they service.'
         WRITE(*,*)' '
         WRITE(*,*)' '
         DO 9 I=1,NQUSE
            WRITE(*,110)I
            WRITE(*,'(A\)')' is serviced by ---------------------> '
            READ(*,*)NSTAND(I)
```

```
          WRITE(*,*)' '
9       CONTINUE
110     FORMAT(1X,'Enter the number of checkstands Queue Line',I2)
        WRITE(*,*)' '
111     WRITE(*,*)'Enter the number of Queue Lines that will be'
        WRITE(*,'(A\)')' open at start of business ------------> '
        READ(*,*)NQUE
        IF(NQUE.GT.NQUSE)THEN
               WRITE(*,*)'The number of Queue Lines open at start of'
               WRITE(*,112)NQUSE
               WRITE(*,*)' '
               CALL ERRER
               CALL BLANK
               GO TO 111
112      FORMAT(1X,'business cannot exceed ',I2)
         ENDIF
         WRITE(*,*)' '
         DO 11 I=1,NQUE
          WRITE(*,120)I
          WRITE(*,'(A\)')' at the start of business----------------> '
          READ(*,*)NSTRT(I)
          WRITE(*,*)' '
11       CONTINUE
120  FORMAT(1X,'How many checkstands will be open for Queue Line ',I2)
         CALL BLANK
         WRITE(*,*)'The following are your inputs ...'
40       WRITE(*,*)' '
         DO 15 I=1,NQUSE
           WRITE(*,130)I,NSTAND(I)
15       CONTINUE
130      FORMAT(1X,'Max # checkstands servicing Queue Line ',I2,' : ',I2)
         WRITE(*,135)NQUE
135      FORMAT(1X,'# Queue Lines open at start of business   : ',I2)
         DO 20 I=1,NQUE
           WRITE(*,140)I,NSTRT(I)
20       CONTINUE
140      FORMAT(1X,'# checkstands open for Queue Line',i2,
    1          ' at start of business   : ',I2)
         WRITE(*,*)' '
         WRITE(*,'(A\)')' Are they correct ? [0-No, 1-Yes] --> '
         READ(*,*)NANSWR
         WRITE(*,*)' '
         WRITE(*,*)' '
         IF(NANSWR.EQ.1)GO TO 10
         IF(NANSWR.EQ.0)THEN
         WRITE(*,*)' '
41       WRITE(*,*)'Which of the following is incorrect ? :'
         WRITE(*,*)' '
         WRITE(*,*)' 1 - Max # checkstands serving a Queue Line'
         WRITE(*,*)' 2 - # Queue Lines open at start of business'
         WRITE(*,*)' 3 - # Checkstands open for a Queue Line at'
         WRITE(*,*)'     start of business'
         WRITE(*,*)' '
```

```
          WRITE(*,'(A\)')' Enter the number of the incorrect value --> '
          READ(*,*)NWRONG
          IF(NWRONG.EQ.1)THEN
            WRITE(*,*)' Which of the following is incorrect ? :'
            WRITE(*,*)' '
            DO 30 I=1,NQUSE
              WRITE(*,150)I,NSTAND(I)
30          CONTINUE
150        FORMAT(1X,I2,' - Max # checkstands serving Queue Line ',I2)
           WRITE(*,*)' '
           WRITE(*,'(A\)')' Enter the number of the incorrect value --> '
           READ(*,*)NWRONG
           WRITE(*,*)' '
           WRITE(*,*)'Re-enter the maximum # of checkstands'
           WRITE(*,'(A\)')' that can service this Queue Line --> '
           READ(*,*)NSTAND(NWRONG)
           CALL BLANK
           WRITE(*,*)' After correction your inputs are now ...'
           GO TO 40
         ELSEIF(NWRONG.EQ.2)THEN
           WRITE(*,*)' '
           WRITE(*,*)'Re-enter the number of Queue Lines open'
           WRITE(*,'(A\)')' at start of business ----------------> '
           READ(*,*)NQUE
           CALL BLANK
           WRITE(*,*)' After corrections your inputs are ...'
           GO TO 40
         ELSEIF(NWRONG.EQ.3)THEN
           WRITE(*,*)' '
           WRITE(*,*)' Which of the following is incorrect ? :'
           WRITE(*,*)' '
           DO 31 I=1,NQUE
             WRITE(*,160)I,I,NSTRT(I)
31         CONTINUE
160     FORMAT(1X,I2,' - # checkstands open for Queue Line ',I2
     1.'    at start of business : ',I2)
            WRITE(*,*)' '
          WRITE(*,'(A\)')' Enter the number of the incorrect value -->'
            READ(*,*)NWRONG
            WRITE(*,'(A\)')' Re-enter the # of checkstands open --> '
            READ(*,*)NSTRT(NWRONG)
            CALL BLANK
            WRITE(*,*)' After correction your inputs are now ...'
            GO TO 40
          ELSE
            CALL ERRER
            CALL BLANK
            GO TO 41
          ENDIF
         ELSE
           CALL ERRER
           CALL BLANK
           WRITE(*,*)' The following are your inputs ...'
```

```
            GO TO 40
          ENDIF
C
C**************************************************************
C
C   After FRONT END information has been entered, Subroutine
C   A R V L is called. The user is then prompted to enter
C   customer arrival rates for each hour of operation.
C
C**************************************************************
C
10       CALL ARVL(NOPEN,NCLOSE)
         CLOSE(2)
         OPEN(2,FILE=FILENAM,STATUS='NEW')
         WRITE(2,*)NPARK,NLRGE,NSMAL,NBAG,NQUSE,NWAIT,
       1 NOPEN,NCLOSE
           WRITE(2,*)NQUE
           DO 75 I=1,NQUSE
              WRITE(2,*)NSTAND(I)
75         CONTINUE
           DO 80 I=1,NQUE
              WRITE(2,*)NSTRT(I)
80         CONTINUE
            NEND=(NCLOSE-NOPEN)/100
            IF(REAL(IDIFF).NE.DIFF)NEND=NEND+1
            DO 70 I=1,NEND
              WRITE(2,*)C(I)
70          CONTINUE
71          CONTINUE
            CLOSE(2)
C
C*************************************************************
C
C This part of the subroutine along with USERF(2) enables
C time between arrivals from a non-stationary Poisson process
C to be generated. Subroutine INTLC calculates the expectation
C function A(t) of an arrival rate function from a
C non-stationary Poisson process defined by the user.  The
C user provides an input file (RATE.IN) of A( t, lambda(t) )
C coordinates representing the arrival rate function
C of the Poisson process.  The user also initializes NPTS (below).
C USERF(1) is subsequently invoked by the user each time a
C time between arrivals is needed.
C
C RATEIN = Array holding data points of Poisson arrival function
C A = Array holding data points of the expectation (integral)
C of RATEIN NPTS = Number of data points in RATEIN
C
C
C
        IF(REAL(IDIFF).NE.DIFF)NPTS=NPTS+1
C       Programmed by:  Capt John Hertz    GOR-87D      Date:  17 Aug 87
C       Modified by:  Capt Roger Moulder                Date: 31 Aug 87
```

```
C
      SSUBN = 0.0
C*****************************************************************
C
C User must set NPTS = # of data points to be read in from RATE.I
      NPTS = (nclose-nopen)/100
C Input user-provided array of ( t,lambda(t) ) coordinates
      DO 100 J = 1,NPTS
         RATEIN(J,1)=J
         RATEIN(J,2)=C(J)
  100 CONTINUE
      TIME = RATEIN(1,1)
C
C*****************************************************************
C Calculate the expectation function A(t) by integrating RATEIN
C using the trapezoid rule to approximate the integral of RATEIN.
C The resulting expectation function data points are stored in the
C array A.  Array A is used by USERF(1).
C
C  N = Number of subintervals (trapezoids)
C  AREA = Area of a single trapezoid (subinterval)
C  WIDTH = Distance between the 2 horizontal coordinates of the trap.
C  AVGHT = Average height of the 2 vertical coordinates of the trap.
C*****************************************************************
C
      N = NPTS - 1
      AREA = 0.0
      A(1,1) = RATEIN(1,1)
      A(1,2) = 0.0
C**** Calculate area for each subinterval (trapezoid) in RATEIN
      DO 300 J = 1,N
         WIDTH = RATEIN(J+1,1) - RATEIN(J,1)
         AVGHT = ( RATEIN(J,2) + RATEIN(J+1,2) ) / 2.0
         AREA  = WIDTH * AVGHT
C****    Accumulate expectation function data points
         A(J+1,2) = A(J,2) + AREA
         A(J+1,1) = RATEIN(J+1,1)
  300 CONTINUE
C
C*****************************************************************
C
C   At this point all user input has been verified as correct.
C   Appropriate calls are made to alter resources.
C
C*****************************************************************
C
          DO 60 K=1,NWAIT
              ATRIB(1)=0.0
              CALL FILEM(2,ATRIB)
60        CONTINUE
       CALL ALTER(1,NPARK)
       CALL SEIZE(1,NWAIT)
       CALL ALTER(2,NSMAL)
```

```
      CALL ALTER(3,NLRGE)
         DO 13 I=5,NQUE+4
             CALL ALTER(I,NSTRT(I-4))
13       CONTINUE
         NLIN=NBAG/NQUE
         IF(NLIN*NQUE.NE.NBAG)THEN
               NLEFT=NBAG-NLIN*NQUE
               DO 7 J=40,NLEFT+39
                 CALL ALTER(J,NLIN+1)
7              CONTINUE
             DO 14 I=J,NQUE+39
               CALL ALTER(I,NLIN)
14           CONTINUE
         ELSE
               DO 8 J=40,NQUE+39
                 CALL ALTER(J,NLIN)
8              CONTINUE
         ENDIF
      IF(NNRUN.EQ.1)THEN
        CALL BLANK
        WRITE(*,'(A\)')' Enter File Name for Output Statistics --> '
        READ(*,'(A12)')FIL2
        OPEN(3,FILE=FIL2,STATUS='NEW')
        WRITE(*,'(A\)')' Enter number of Runs desired --> '
        READ(*,*)XX(28)
        OPEN(8,FILE='RUN.TMP',STATUS='NEW')
        WRITE(8,*)XX(28)
        CLOSE(8)
      ELSE
        OPEN(8,FILE='RUN.TMP',STATUS='OLD')
        READ(8,*)XX(28)
        CLOSE(8)
      ENDIF
         CALL BLANK
      WRITE(*,400)NNRUN,INT(XX(28))
400   FORMAT(1X,'Run ',I2,' of ',I2)
      WRITE(*,*)' '
         WRITE(*,*)'Simulation Begins ...'
      WRITE(*,*)' '
      RETURN
      END
C
C
C
      SUBROUTINE EVENT(I)
C
***********************************************************
C
C This Subroutine has twelve events which are called as the
C simulation is running. They are as follows:
C
C   EVENT 1 - This event finds the shortest Queue line
C             to place a customer in after the customer
```

```
C             has finished shopping.
C
C   EVENT 2 - This event allocates the number of cashiers
C             that will service a Queue line.
C
C   EVENT 3 - This event signals the user that no more
C             customers will enter the commissary.
C
C   EVENT 4 - This event calculates the average waiting
C             times of customers in the Queue lines.
C
C   EVENT 5 - This event opens or closes an express line
C             as needed.
C
C   EVENT 6 - This event alters the cashier and bagger
C             resources of a Queue Line that has been
C             closed down. Before the resources can be
C             altered, the customers in the Queue at the
C             time it is closed must be serviced.
C             Express Line.
C
C   EVENT 7 - This event changes the value of the congestion
C             factor to reflect the store has low level
C             congestion.
C
C   EVENT 8 - This event changes the congestion factor to
C             reflect there is no congestion present.
C
C   EVENT 9 -  This event changes the value of the congestion
C              factor to feflect the store has medium level
C              congestion.
C
C   EVENT 10 - This event changes the value of the congestion
C              factor to reflect the store has high level
C              congestion.
C
C   EVENT 11 - This event determines the number of items each
C              customer buys. If the number of items is more
C              than 100,then the customer will need 2 carts.
C              Atribute 3 is set equal to the number of carts
c              each customer needs.
C
C   EVENT 12 - This event determines the number of items each
C              express customer buys.
C***************************************************************
C
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
       COMMON/USER1/NQUE,NOPT,NSTAND(38),NSTRT(38),NQUSE,C(38)
       NIS=0
       DO 13 J=7,NQUE+6
            NIS=NIS+NNQ(J)
13     CONTINUE
```

```
          DO 14 J=1,82
            NIS=NIS+NNACT(J)
14       CONTINUE
         XX(6)=REAL(NIS)
         II=0
         DO 55 J=1,NQUE
           II=II+NNRSC(J+4)+NRUSE(J+4)
55       CONTINUE
         GO TO(1,2,3,4,5,6,7,8,9,10,11,12)I
C
C*****************************************************************
C
C   EVENT 1 finds the shortest Queue Line and places the customer
C   in that line.
C
C*****************************************************************
C
1        TEMP=99999.0
           DO 30 J=7,NQUE+6
             WAIT= (NNQ(J))/((NNRSC(J-2)+NRUSE(J-2))*(1.0/XX(5)))
             IF(WAIT.LT.TEMP)THEN
                 NSHORT=J
                 TEMP=WAIT
             ENDIF
30       CONTINUE
           CALL FILEM(NSHORT,ATRIB)
           RETURN
C
C****************************************************************
C
C    EVENT 2 removes cashiers and closes Queue Lines as needed.
C
C
C****************************************************************
C
2        ATRIB(1)=ATRIB(1)+XX(32)
           IF(TNOW.LT.XX(1))CALL SCHDL(2,XX(32),ATRIB)
           ITIME=INT(ATRIB(1)/60.0)
C
C****************************************************************
C
C   This part of the code checks to see if waiting time in line
C   will be greater than 15 minutes. If so, then an attempt will
C   be made to open another checkstand.
C
C****************************************************************
C
         IF(TNOW.GE.XX(32))WRITE(*,100)ITIME,NIS,II,XX(13)
           IWAIT=0
           IALT=0
           DO 20 I=5,NQUE+4
               WAIT=(NNQ(I+2))/((NNRSC(I)+NRUSE(I))*(1.0/XX(5)))
               IF(WAIT.GT.15.0)THEN
```

```
                IWAIT=1
                IF(NNRSC(I)+NRUSE(I).LT.NSTAND(I-4))THEN
                  IALT=1
                  CALL ALTER(I,1)
                  CALL ALTER(I+35,5)
                ENDIF
              ENDIF
20          CONTINUE
C
****************************************************************
C
C     If all checkstands were open for a queue line, an attempt
C     is made to open another queue line.
C
****************************************************************
C
          IF(IWAIT.EQ.1)THEN
           IF(IALT.NE.1)THEN
            IF(NQUE+1.LE.NQUSE)THEN
             IF(NNRSC(NQUE+5)+NRUSE(NQUE+5).LT.NSTAND(NQUE+1))THEN
                  NQUE=NQUE+1
                  CALL ALTER(NQUE+4,1)
                  CALL ALTER(NQUE+39,5)
                  ENDIF
                ENDIF
            ENDIF
          ENDIF
c
C*******************************************************************
C
C     This portion of the subroutine checks to see if a cashier or
C     queue line should be removed.
C
C*******************************************************************
C
          IF(ITIME.GT.1)THEN
            DO 45 J=NQUE+4,5,-1
                WAIT=(NNQ(J+2))/((NNRSC(J)+NRUSE(J))*(1.0/XX(5)))
           IF(WAIT.LT.15.0.AND.C(ITIME).LT.C(ITIME-1))THEN
            IF(ILAST.NE.ITIME)THEN
             IF(NNRSC(J)+NRUSE(J).EQ.1.AND.NNQ(J+2).EQ.0)THEN
                       NQUE=NQUE-1
                       CALL ALTER(NQUE+5,-1)
                       CALL ALTER(NQUE+40,-5)
                       ICHEK=1
             ELSEIF(NNRSC(J)+NRUSE(J).EQ.1.AND.NNQ(J+2).GT.0)THEN
                       NQUE=NQUE-1
                       FIN=XX(5)*(REAL(NNQ(J+2)))
                       ATRIB(1)=REAL(J)
                       CALL SCHDL(6,FIN,ATRIB)
                       ICHEK=1
             ELSE
                       CALL ALTER(J,-1)
```

```
                        CALL ALTER(J+35,-5)
                        ICHEK=1
                  ENDIF
                     SUM=0.0
                     DO 47 LL=5,NQUE+4
                        SUM=SUM+NNRSC(LL)+NRUSE(LL)
47                   CONTINUE
                     IF(NQUE.EQ.INT(SUM))GO TO 46
               ENDIF
              ENDIF
45             CONTINUE
46             CONTINUE
              IF(ICHEK.EQ.1)ILAST=ITIME
           ENDIF
           RETURN
C
C**************************************************************
C
C EVENT 3 signals the user that no more customers can enter the
C the Commissary. Those customers in the store at closing time
C will be processed until all have been serviced.
C
C**************************************************************
C
3          WRITE(*,*)' '
           WRITE(*,*)' '
           WRITE(*,*)' Closing Front Door ...'
           WRITE(*,*)' '
           WRITE(*,*)' '
           RETURN
C
C*********************************************************
C
C EVENT 4 calculates the average waiting time of customers
C in the checkout lines.
**********************************************************
4          IF(NIS.NE.0)CALL SCHDL(4,XX(32),ATRIB)
           IF(NIS.EQ.0)WRITE(10,*)XX(33)/XX(34)
              ICNT=13
              TOT2=0.0
              NOBS2=0
              DO 40 I=7,NQUE+6
                    TOT2=TOT2+FFAWT(I)*NNCNT(ICNT)
                    NOBS2=NOBS2+NNCNT(ICNT)
                    ICNT=ICNT+2
40            CONTINUE
              IF(NOBS2.LE.0.OR.NOBS2.EQ.INT(XX(15)))THEN
              XX(13)=0.0
              GO TO 41
           ENDIF
              XX(13)=(TOT2-XX(14))/(NOBS2-INT(XX(15)))
              IF(XX(13).LT.0.0)XX(13)=0.0
              XX(14)=TOT2
```

```
          XX(15)=NOBS2
41      IF(TNOW.LE.XX(1))THEN
          WRITE(4,*)XX(13),II,TNOW
          XX(30)=XX(30)+1.0
        ENDIF
        IF(TNOW.GT.XX(1))THEN
          XX(33)=XX(33)+XX(13)
          XX(34)=XX(34)+1.0
        ENDIF
        RETURN
C**************************************************
C
C         EVENT 5 - Adds or deletes an Express Line
C
C**************************************************
5      ATRIB(1)=ATRIB(1)+XX(32)
       IF(TNOW.LT.XX(1))CALL SCHDL(5,XX(32),ATRIB)
       WAIT=(NNQ(6))/((NNRSC(4)+NRUSE(4))*(1.0/2.0))
       IF(ATRIB(1)-XX(16).GE.30.0)THEN
        IF(WAIT.GT.5.0.AND.NNRSC(4)+NRUSE(4).LT.2)CALL ALTER(4,1)
        IF(WAIT.LT.5.0.AND.NNRSC(4)+NRUSE(4).EQ.2)CALL ALTER(4,-1)
        XX(16)=ATRIB(1)
       ENDIF
       RETURN
C
C***********************************************************
C
C       EVENT 6 - Removes a cashier from a 1 to 1 queue line
C                 after all customers have been serviced.
C
C***********************************************************
C
6          LOC=INT(ATRIB(1))
           IF(NNQ(NQUE+7).EQ.0)THEN
                 CALL ALTER(NQUE+5,-1)
                 CALL ALTER(NQUE+40,-5)
           ELSE
                 CALL SCHDL(6,5.0,ATRIB)
           ENDIF
           RETURN
C
*********************************************************
C
C    EVENT 7 changes the congestion factor to reflect low
C    congestion.
C
*********************************************************
C
7       XX(9)=1.1
        RETURN
C
*********************************************************
C
```

```
C    EVENT 8 changes the congestion factor to reflect
C    crowding has no effect on shopping time.
C
C*********************************************************
C
8         XX(9)=1.0
          RETURN
C
C*********************************************************
C
C     EVENT 9 changes the congestion factor to reflect
C     medium congestion.
C
C*********************************************************
C
9        XX(9)=1.2
         RETURN
C
C*********************************************************
C
C     EVENT 10 changes the congestion factor to reflect
C     high congestion.
C
C*********************************************************
C
10       XX(9)=1.4
         RETURN
C
C*********************************************************
C
C     EVENT 11 assigns the number of items each customer
C     buys. Attribute 3 is set to the number of carts the
C     customer will need. If the customer buys 100 or
C     more they will need 2 carts.
C
C*********************************************************
11       ATRIB(2)=GAMA(23.1,3.2,2)
         ATRIB(3)=1.0
         IF(TNOW.GT.(XX(1)-45.0))THEN
           IF(ATRIB(2).GT.70.0)ATRIB(2)=70
         ENDIF
         IF(ATRIB(2).GT.100.0)ATRIB(3)=2.0
         RETURN
C
C*********************************************************
C
C    EVENT 12 assigns the number of items each express
C    customer buys.
C
C*********************************************************
C
12       ATRIB(2)=UNFRM(1.0,15.0,7)
         RETURN
```

```
100    FORMAT(1X,'Hour:',I2,'   No. in Store:',I4,'   No. Cashiers: '
     1,I2,'  Avg. Wait: ',F6.2)
       END
C
C
C
       SUBROUTINE BLANK
C
**********************************************************
C
C     This subroutine is used to clear the screen at
C     different times so that the user can better respond
C     to prompts.
C
C*********************************************************
C
       DO 10 I=1,25
         WRITE(*,*)' '
 10    CONTINUE
       RETURN
       END
C
C
C
**********************************************************
C
C    This subroutine is called whenever the user has
C    responded incorrectly to a prompt. Control is
C    returned to the calling subroutine and the user is
C    prompted to re-enter their input.
C
**********************************************************
C
          SUBROUTINE ERRER
          CHARACTER*2 DUMMY
          WRITE(*,*)' '
          WRITE(*,*)'Invalid response ...'
          WRITE(*,'(A\)')' Hit RETURN to continue --> '
          READ(*,'(A1)')DUMMY
          RETURN
          END
C
C
C
        FUNCTION USERF(I)
C
**********************************************************
C
C  This Function has two purposes as follows :
C
C   FUNCTION 1 - Calculates the time between each customer
C                arrival.
C
```

```
C   FUNCTION 2 - Calculates each customers' shopping time.
C
C   FUNCTION 3 - Calculates each regualr customer's checkout
C                time.
C
C   FUNCTION 4 - Calculates each express customer's checkout
C                time.
C
C
C************************************************************
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/NSTPSN/A(1000,2),RATEIN(1000,2),SSUBN,NPTS,TIME
      INTEGER LOW,HIGH,MID
      GO TO (1,2,3,4)I
C
C************************************************************
C
C   FUNCTION 1 calculates the time between customer arrivals.
C
C************************************************************
    1 CONTINUE
C**** Given the expectation function expressed as data points
C     in array A, USERF(1) generates the time before next
C     arrival in the Poisson process. USERF(1) should be
C     invoked by the user each time a time between arrivals
C     is required.  The next arrival time is generated using
C     an algorithm taken from 'Introduction to Stochastic
C     Processes' by Erhan Cinlar, page 99.
      LOW = 1
      HIGH = NPTS
      RANNUM = DRAND(8)
      XSUBN = - ALOG(RANNUM)
C**** Start arrival process cycle over again if SSUBN not
C     in range
      IF (SSUBN+XSUBN .GT. A(HIGH,2)) THEN
         SSUBN = ( (SSUBN + XSUBN) - A(HIGH,2) ) + A(LOW,2)
         TIME = RATEIN(1,1)
      ELSE
         SSUBN = SSUBN + XSUBN
      ENDIF
C**** Use binary search to locate vertical interval in A
C     where SSUBN
   50 IF (HIGH-LOW.LE.1) GO TO 60
         MID = IFIX( FLOAT(HIGH+LOW) / 2.0 )
         IF (SSUBN.LE.A(MID,2)) THEN
            HIGH = MID
         ELSE
            LOW = MID
         ENDIF
         GO TO 50
C**** Calculate arrival time using the approximating
C**** relationship y = mx + b between pairs of data
```

```
C**** points in A
   60 SLOPE = ( A(HIGH,2)-A(LOW,2) ) / ( A(HIGH,1)-A(LOW,1) )
      B = A(HIGH,2) - SLOPE * A(HIGH,1)
C**** Next arrival time = (y - b) / m
      X = (SSUBN - B) / SLOPE
C**** Time duration between last arrival and next arrival
      TBA = X - TIME
C**** Save next arrival time
      TIME = X
C**** Return duration until next arrival
      USERF = TBA
      RETURN
C
**********************************************************
C
C   FUNCTION 2 calculates each regular customer's shopping
C   time.
C
C*********************************************************
C
2       USERF=(XX(17)+XX(18)*(INT(ATRIB(2)))+RNORM(0,XX(23),3))*XX(9)
        IF(TNOW.GT.(XX(1)-45.0))THEN
          USERF=(XX(18)*(INT(ATRIB(2)))+RNORM(0,XX(23),3))*XX(9)
        ENDIF
        RETURN
C
C*********************************************************
C
C    FUNCTION 3 calculates the checkout for regular customers
C    based on the number of items bought.
C
C*********************************************************
C
3        USERF=XX(19)+XX(20)*(INT(ATRIB(2)))+RNORM(0,XX(24),4)
        IF(TNOW.GT.XX(1))USERF=XX(20)*(INT(ATRIB(2)))+RNORM(0,XX(24),4)
         RETURN
C
C*********************************************************
C
C     FUNCTION 4 calculates the checkout time for express
C     customers based on the number of items bought.
C
C*********************************************************
C
4        USERF=(XX(21)+XX(22)*(INT(ATRIB(2)))+RNORM(0,XX(25),5))*XX(9)
         RETURN
         END
C
C
C
         SUBROUTINE ARVL(NOPEN,NCLOSE)
C
**********************************************************
```

```
C
C This Subroutine prompts the user for arrival rate data.
C
************************************************************
C
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
       COMMON/USER1/NQUE,NOPT,NSTAND(38),NSTRT(38),NQUSE,C(38)
       DIMENSION A(4),B(4)
       CHARACTER*2 DUMMY
       CALL BLANK
       IDIFF=NCLOSE-NOPEN
       IDIV=IDIFF/100
       IMULT=IDIV*100
       IF(NOPT.EQ.2)THEN
         WRITE(*,*)' The following will be used as inputs ...'
                NEND=(NCLOSE-NOPEN)/100
                IF(IDIFF.NE.IMULT)NEND=NEND+1
                DO 5 I=1,NEND
                   READ(2,*)C(I)
5               CONTINUE
                GO TO 1
       ENDIF
       IF(NOPT.EQ.4)THEN
         CALL BLANK
         WRITE(*,*)' Since you changed either the OPENING'
         WRITE(*,*)' or CLOSING time, you must re-enter'
         WRITE(*,*)' all arrival rates'
         WRITE(*,*)'  '
         WRITE(*,'(A\)')' Hit RETURN to continue --> '
         READ(*,'(A1)')DUMMY
         CALL BLANK
       ENDIF
         WRITE(*,*)' '
         WRITE(*,*)' You will now be prompted to enter '
         WRITE(*,*)' Customer arrival rates per minute ...'
         WRITE(*,*)' '
         K=1
         DO 10 I=NOPEN,NCLOSE-100,100
           WRITE(*,100)I,I+100
           WRITE(*,*)' '
           WRITE(*,'(A\)')' Enter customer arrival rate -------> '
           READ(*,*)C(K)
           WRITE(*,*)' '
           K=K+1
10       CONTINUE
         IF(IDIFF.NE.IMULT)THEN
         WRITE(*,100)I,NCLOSE
         WRITE(*,*)' '
         WRITE(*,'(A\)')' Enter customer arrival rate -------> '
         READ(*,*)C(K)
       ENDIF
       CALL BLANK
```

```
2       WRITE(*,*)' The following are your inputs ...'
1       WRITE(*,*)' '
           WRITE(*,*)' FROM  TO    ARRIVAL RATE'
           WRITE(*,*)' ----  ----  ------------'
           K=1
           DO 20 I=NOPEN,NCLOSE-100,100
                 WRITE(*,110)I,I+100,C(K)
                 K=K+1
20         CONTINUE
           IF(IDIFF.NE.IMULT)WRITE(*,110)I,NCLOSE,C(K)
           WRITE(*,*)' '
           WRITE(*,'(A\)')' Are they correct ? [0-No, 1-Yes] --> '
           READ(*,*)NANSWR
           IF(NANSWR.EQ.1)THEN
             TIM=0.0
             RETURN
           ELSEIF(NANSWR.EQ.0)THEN
             CALL BLANK
             K=1
             WRITE(*,*)' Which of the folowing is incorrect ? :'
             WRITE(*,*)' '
             WRITE(*,*)'    FROM  TO    ARRIVAL RATE'
             WRITE(*,*)'    ----  ----  ------------'
             DO 40 I=NOPEN,NCLOSE-100,100
               WRITE(*,120)K,I,I+100,C(K)
               K=K+1
40           CONTINUE
             IF(IDIFF.NE.IMULT)WRITE(*,120)K,I,NCLOSE,C(K)
             WRITE(*,*)
             WRITE(*,'(A\)')' Enter the incorrect value by number--> '
             READ(*,*)NWRONG
             WRITE(*,'(A\)')' Re-enter the arrival rate --------> '
             READ(*,*)C(NWRONG)
             CALL BLANK
             WRITE(*,*)' After  correction your inputs are now ...'
             GO TO 1
           ELSE
             WRITE(*,*)' '
             WRITE(*,*)' '
             CALL ERRER
             CALL BLANK
             GO TO 2
           ENDIF
110     FORMAT(2X,I4,2X,I4,6X,F4.2)
120     FORMAT(1X,I2,'- ',I4,2X,I4,6X.
100     FORMAT(1X,I4,' Hrs To ',I4,' Hr      )
           RETURN
           END
C
C
C
        SUBROUTINE OTPUT
C
```

```
C***********************************************************
C
C  This SUBROUTINE tailors the output statistics of a
C  simulation run based on the number of Queue Lines of
C  the store being simulated.
C
C***********************************************************
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/USER1/NQUE,NOPT,NSTAND(38),NSTRT(38),NQUSE,C(38)
      COMMON/USER2/NOPEN,NCLOSE
      DIMENSION XSTORE(200),ISTORE(200),XTIM(200),XOUT(100),XIOUT(100)
      DIMENSION CI1L(100),CI1U(100),CI2L(100),CI2U(100),IHRS(100),T(22)
      DIMENSION XAFTER(22)
      CHARACTER*12 FILE1,FILE2
      CHARACTER*4 NAME
      CHARACTER*2 DUMMY
      T(1)=12.706
      T(2)=4.303
      T(3)=3.182
      T(4)=2.776
      T(5)=2.571
      T(6)=2.447
      T(7)=2.365
      T(8)=2.306
      T(9)=2.262
      T(10)=2.228
      T(11)=2.201
      T(12)=2.179
      T(13)=2.160
      T(14)=2.145
      T(15)=2.131
      T(16)=2.120
      T(17)=2.110
      T(18)=2.101
      T(19)=2.093
      T(20)=2.086
      WRITE(3,*)' '
      ISUB=INT(TNOW)-INT(XX(1))
      IREMAN=MOD(ISUB,60)
      IADD=((ISUB/60)*100)+IREMAN
      ICURNT=NCLOSE+IADD
      WRITE(3,300)NNRUN,INT(XX(28))
      WRITE(3,*)' '
300   FORMAT(1X,'Run ',I2,' of ',I2)
      WRITE(3,5)ICURNT
5     FORMAT(1X,'CURRENT TIME ',I4)
      WRITE(3,*)' '
      WRITE(3,*)' '
      WRITE(3,*) '           **FILE STATISTICS**'
      WRITE(3,*) ' '
      WRITE(3,*)' FILE  NODE   AVERAGE    AVERAGE    STANDARD   MAX'
      WRITE(3,*)' NO.   LABEL  LENGTH     WAIT TIME  DEVIATION  LENGTH'
```

```
      WRITE(3,*)' ----  -----  ---------  ---------  ---------  ------'
      DO 10 J=1,NQUSE+6
       IF(J.EQ.1)NAME='PKNG'
       IF(J.EQ.2)NAME='DESK'
       IF(J.EQ.3)NAME='CHEK'
       IF(J.EQ.4)NAME='REG '
       IF(J.EQ.5)NAME='XPRS'
       IF(J.EQ.6)NAME='XLIN'
       IF(J.GT.6)NAME='LIN'
       IF(J.LE.6)WRITE(3,15)J,NAME,FFAVG(J),FFAWT(J),FFSTD(J),FFMAX(J)
       IF(J.GT.6)WRITE(3,16)J,NAME,J-6,FFAVG(J),FFAWT(J),FFSTD(J),
     1  FFMAX(J)
10    CONTINUE
      WRITE(3,*)' '
      WRITE(3,*)' '
      WRITE(3,*)' **STATISTICS FOR VARIABLES BASED ON OBSERVATION**'
      WRITE(3,*)' '
        WRITE(3,*)'VARIABLE    MEAN    STD     MIN     MAX     NO.OF'
        WRITE(3,*)'NAME        VALUE   DEV     VALUE   VALUE   OBS'
        WRITE(3,*)'----------  ------  ------  ------  ------  ------'
      WRITE(3,27)CCAVG(1),CCSTD(1),CCMIN(1),CCMAX(1),CCNUM(1)
      DO 20 K=1,NQUSE+1
      IF(K.EQ.1)WRITE(3,25)CCAVG(K+5),CCSTD(K+5),CCMIN(K+5),
     1CCMAX(K+5),CCNUM(K+5)
      IF(K.GT.1)WRITE(3,26)K-1,CCAVG(K+5),CCSTD(K+5),CCMIN(K+5),
     2CCMAX(K+5),CCNUM(K+5)
20    CONTINUE
      WRITE(3,*)' '
      WRITE(3,*)' '
      WRITE(3,*)'       **RESOURCE STATISTICS**'
      WRITE(3,*)' '
      WRITE(3,*)'RESOURCE   RESOURCE   AVERAGE  STANDARD MAXIMUM'
      WRITE(3,*)'LABEL      UTIL       AVAIL    DEV      UTIL'
      WRITE(3,*)'--------   --------   -------  -------- -------'
      DO 30 L=1,NQUSE+4
       IF(L.EQ.1)NAME='PARK'
       IF(L.EQ.2)NAME='SMAL'
       IF(L.EQ.3)NAME='LRGE'
       IF(L.EQ.4)NAME='XCSH'
       IF(L.GT.4)NAME='CASH'
       IF(L.LE.4)WRITE(3,35)NAME,RRAVG(L),RRAVA(L),RRSTD(L),RRMAX(L)
       IF(L.GT.4)WRITE(3,36)NAME,L-4,RRAVG(L),RRAVA(L),RRSTD(L),
     1RRMAX(L)
30    CONTINUE
      NAME='BAG'
      DO 50 J=40,NQUSE+39
         WRITE(3,36)NAME,J-39,RRAVG(J),RRAVA(J),RRSTD(J),RRMAX(J)
50    CONTINUE
      WRITE(3,*)' '
      WRITE(3,*)' '
      WRITE(3,*)'     **ACTIVITY STATISTICS**'
      WRITE(3,*)' '
      WRITE(3,*)' ACTIVITY  AVERAGE  MAXIMUM  STANDARD   ENTITY'
```

```
      WRITE(3,*)' INDEX     UTIL     UTIL     DEVIATION  COUNT'
      WRITE(3,*)' --------  -------  -------  ---------  ------'
      NEND=NQUSE*2
      DO 40 M=1,NEND+12
        WRITE(3,45)M,AAAVG(M),AAMAX(M),AASTD(M),NNCNT(M)
40    CONTINUE
      IF(XX(28).EQ.1.0)STOP
      IF(INT(XX(28)).LE.NNRUN)THEN
        NEND=INT(XX(30))*NNRUN
        REWIND(10)
        AFTSUM=0.0
        DO 240 K=1,NNRUN
          READ(10,*)XAFTER(K)
          AFTSUM=AFTSUM+XAFTER(K)
240     CONTINUE
        AFTSUM=AFTSUM/(REAL(NNRUN))
        VAR2=0.0
        DO 250 K=1,NNRUN
          VAR2=VAR2+(XAFTER(K)-AFTSUM)**2
250     CONTINUE
        VAR2=VAR2/(REAL(NNRUN-1))
        STDEV2=SQRT(VAR2/REAL(NNRUN))
        DELTA2=STDEV2*T(NNRUN-1)
        AFTL=AFTSUM-DELTA2
        AFTU=AFTSUM+DELTA2
          REWIND(4)
        DO 200 J=1,NEND
          READ(4,*)XSTORE(J),ISTORE(J),XTIM(J)
200     CONTINUE
        NN=INT(XX(30))
        DO 210 J=1,NN
          IP=0
          XSUM=0.0
          ISUM=0
          ILO=999
          IHI=0
          DO 220 K=1,NNRUN
            XSUM=XSUM+XSTORE(J+IP)
            ISUM=ISUM+ISTORE(J+IP)
            IF(ISTORE(J+IP).LT.ILO)ILO=ISTORE(J+IP)
            IF(ISTORE(J+IP).GT.IHI)IHI=ISTORE(J+IP)
            IP=IP+NN
220       CONTINUE
          XX(26)=XSUM/REAL(NNRUN)
          XX(27)=REAL(ISUM)/REAL(NNRUN)
          XOUT(J)=XX(26)
          XIOUT(J)=XX(27)
          VAR1=0.0
          IP=0
          DO 230 M=1,NNRUN
            VAR1=VAR1+(XSTORE(J+IP)-XX(26))**2
            IP=IP+NN
230       CONTINUE
```

```
          VAR1=VAR1/REAL(NNRUN-1)
          STDEV1=SQRT(VAR1/REAL(NNRUN))
          DELTA=STDEV1*T(NNRUN-1)
          CI1L(J)=XX(26)-DELTA
          CI1U(J)=XX(26)+DELTA
          IF(CI1L(J).LT.0.0)CI1L(J)=0.0
          IF(CI1U(J).LT.0.0)CI1U(J)=0.0
          CI2L(J)=REAL(ILO)
          CI2U(J)=REAL(IHI)
210     CONTINUE
        CALL BLANK
        ILEFT=(NOPEN/100)*100
        IBEGIN=NOPEN-ILEFT
        ITIM=IBEGIN
        INCR=0
        DO 540 I=1,NN
          ITIM=ITIM+INT(XX(32))
          IF(ITIM.EQ.60)THEN
            INCR=INCR+100
            ITIM=0
          ENDIF
          IPLUS=INCR+ITIM
          IHRS(I)=ILEFT+IPLUS
540     CONTINUE
        WRITE(*,'(A\)')' Enter File Name for Wait Time Plot --> '
        READ(*,'(A12)')FILE1
        WRITE(*,'(A\)')' Enter File Name for # Cashiers Plot --> '
        READ(*,'(A12)')FILE2
        CALL BLANK
        WRITE(*,*)' Writing Plot data to files ...'
        WRITE(*,*)' '
        OPEN(8,FILE=FILE1,STATUS='NEW')
        OPEN(9,FILE=FILE2,STATUS='NEW')
        WRITE(3,*)' '
        WRITE(3,*)'         Average Waiting Time In Queue Line'
        WRITE(3,*)' '
        WRITE(3,*)'                   95% Confidence Interval'
        WRITE(3,*)' Mean          Lower          Upper'
        WRITE(3,*)' Estimate      Limit          Limit          Time'
        WRITE(3,*)' --------      ----------     ----------     ----'
        DO 400 L=1,NN
          WRITE(8,*)XOUT(L),CI1L(L),CI1U(L),IHRS(L)
          WRITE(3,450)XOUT(L),CI1L(L),CI1U(L),IHRS(L)
400     CONTINUE
         WRITE(3,*)' '
         WRITE(3,*)' '
         WRITE(3,415)
415      FORMAT(7X,' Average Waiting Time after Closing')
         WRITE(3,*)' '
         WRITE(3,*)'                  95% Confidence Interval'
         WRITE(3,*)' Mean         Lower          Upper'
         WRITE(3,*)' Estimate     Limit          Limit'
         WRITE(3,*)' --------     ----------     ----------'
```

```
       WRITE(3,460)AFTSUM,AFTL,AFTU
      WRITE(3,*)' '
      WRITE(3,*)'         Average Number of Cashiers on Duty'
      WRITE(3,*)' '
      WRITE(3,*)' Mean         Low           High'
      WRITE(3,*)' Estimate     Value         Value          Time'
      WRITE(3,*)' --------     ----------    ----------     ----'
      DO 410 L=1,NN
        WRITE(9,*)XIOUT(L),CI2L(L),CI2U(L),IHRS(L)
        WRITE(3,450)XIOUT(L),CI2L(L),CI2U(L),IHRS(L)
410   CONTINUE
      CLCSE(4,STATUS='DELETE')
      CLOSE(10,STATUS='DELETE')
      OPEN(10,FILE='RUN.TMP',STATUS='OLD')
      CLOSE(10,STATUS='DELETE')
      CLOSE(3)
      CLOSE(8)
      CLOSE(9)
      STOP
      ENDIF
460   FORMAT(1X,F8.2,5X,F8.2,6X,F8.2)
450   FORMAT(1X,F8.2,5X,F8.2,6X,F8.2,8X,I4)
45    FORMAT(1X,I4,6X,F7.3,2X,F7.3,2X,F7.3,5X,I5)
35    FORMAT(1X,A4,7X,F6.2,4X,F7.2,3X,F6.2,3X,F7.2)
25    FORMAT(1X,'EXPRS TIS',3X,F6.2,2X,F6.2,2X,F6.1,2X,F6.1,2X,F6.1)
15    FORMAT(1X,I3,4X,A4,2X,F9.3,2X,F9.3,2X,F9.3,3X,F6.1)
16    FORMAT(1X,I3,4X,A3,I2,1X,F9.3,2X,F9.3,2X,F9.3,3X,F6.1)
26    FORMAT(1X,'LINE',I2,' TIS',2X,F6.2,2X,F6.2,2X,F6.1,2X,F6.1,2X,
     1F6.1)
27    FORMAT(1X,'PARK BALKS',2X,F6.2,2X,F6.2,2X,F6.2,2X,F6.2,2X,F6.1)
36    FORMAT(1X,A4,I2,5X,F6.2,4X,F7.2,3X,F6.3,3X,F7.2)
      RETURN
      END
```

Appendix B : General Purpose Commissary Store Simulation Model (GPCSS) User's Manual

1. Introduction

This document describes the use of the General Purpose Commissary Store Simulation Model (GPCSS). The model was developed using the SLAM II Simulation language and FORTRAN and is intended for use on the Zenith Z-248 computer system.

It is assumed that users are familiar with MS-DOS and the Zenith Z-248 computer system. For more information regarding MS-DOS or the Z-248 computer system, see the appropriate users' manuals supplied with the computer system. It is also assumed that users will become familiar with the SLAM II PC Version User's Manual, by Pritsker and Associates, Inc. This manual is supplied with the PC SLAM II software.

The GPCSS is composed of four files on the supplied diskette. The files are as follows:

GPCSS.DAT
GPCSS.FOR
GPCSS.TRA
GPCSS.EXE

GPCSS.DAT is the SLAM network source code, and GPCSS.FOR is the FORTRAN subroutines source code. GPCSS.TRA is the translated SLAM network and was obtained by following the instructions in Section 2.2 of the PC SLAM II User's Manual. GPCSS.EXE is the customized execution module obtained by following the instructions is Section 3.3 of the PC SLAM II User's Manual. It is recommended that back-up copies of these

four files be made to prevent loss of the programs in case of accidental erasure.

2. Modifying the SLAM Network and FORTRAN Subroutines

The source code files GPCSS.DAT and GPCSS.FOR have been included on the diskette so that changes to the model can be made if needed. It is recommended that the MS-DOS text editor EDLIN be used to make these changes. Instructions for using EDLIN are covered in Chapter 12 of the MS-DOS manual.

Data collection must be accomplished as outlined in Chapter IV of the thesis, '<u>Development of a General Purpose Commissary Store Simulation Model</u>", by Captain Roger D. Moulder, before parameter values can be entered into the model. Many parameter values are entered interactively as will be described in Section 4 of this manual. Other values must be entered by editing the source code files GPCSS.DAT and GPCSS.FOR. Instructions for making these changes are presented next.

2.1 Changing GPCSS.DAT

GPCSS.DAT is the source code file which contains the SLAM Network statements. Before the model can be used to simulate a commissary, data collection must be accomplished as outlined in the above mentioned thesis. Once the data is collected, the user can edit the file GPCSS.DAT and enter values for the following variables in the SLAM Network :

XX(2) - PERCENT OF CHECK CUSTOMERS
XX(3) - TIME TO STAMP A CHECK
XX(4) - PERCENT EXPRESS CUSTOMERS
XX(5) - MEAN CHECKOUT TIME
XX(6) - INITIAL CONGESTION FACTOR
XX(7) - LOW CONGESTION THRESHOLD
XX(10) - MEDIUM CONGESTION THRESHOLD
XX(11) - HIGH CONGESTION THRESHOLD
XX(17) - Y-INT OF REGULAR SHOPPING TIME
XX(18) - SLOPE OF REGULAR SHOPPING TIME
XX(19) - Y-INT OF REGULAR CHECKOUT TIME
XX(20) - SLOPE OF REGULAR CHECKOUT TIME
XX(21) - Y-INT OF EXPRESS CHECKOUT TIME
XX(22) - SLOPE OF EXPRESS CHECKOUT TIME
XX(23) - MSE OF REGULAR SHOPPING TIME
XX(24) - MSE OF REGULAR CHECKOUT TIME
XX(25) - MSE OF EXPRESS CHECKOUT TIME
XX(32) - TIME BETWEEN SYSTEM STATUS CHECKS

After the changes are made, the SLAM Network must be translated as described in Section 2.2 of the PC SLAM II User's Manual.

2.2 Changing GPCSS.FOR

It may be necessary to change the probability distributions in SUBROUTINE EVENT. These distributions are located in events 11 and 12. Event 11 assigns the number of items bought by regular customers and Event 12 assigns the number of items bought by express customers. As discussed in the thesis cited above, data collection and analysis are necessary so that the distributions for these two events can be determined. After the distributions are determined, the file GPCSS.FOR must be edited and the events changed to the correct distribution types. Line number 888 in GPCSS.FOR is the event for regular customers and line number 903 is for express customers. These distributions are currently set to Gamma(23.1,3.2) and Uniform(7.5,16.3) respectively.

Other changes to the FORTRAN code can be made by the user as deemed appropriate.

After changes have been made, it will be necessary to compile and link the GPCSS.FOR file. Instructions for these procedures are include in the MS-DOS FORTRAN Compiler User's Manual.

After the file has successfully compiled and linked, the model is ready for execution as described in the Section 4 of this manual.

3. Getting Started

The preparation to use the GPCSS Model will be determined by whether the Z-248 computer system to be used has dual floppy disk drives or one floppy disk drive and a hard disk drive. Each of these cases is discussed next.

3.1 Dual Floppy System

To use the model on a dual floppy system, insert the PC SLAM master diskette into drive A. Insert the diskette with the GPCSS files into drive B. Set the default drive to B. The model is now ready for execution as described in Section 4 of this manual. A complete example of model execution is given in Section 5 of this manual.

3.2 Hard Drive System

It is recommended that a di. 'ctory be created for the GPCSS Model and all of the GPCSS iles copied into the

directory. This can be accomplished by the following MS-DOS commands :

```
C>MKDIR COMMSIM
C>CD COMMSIM
C>COPY A:*.*
```

Before typing in the third command, be sure the diskette with the GPCSS files has been inserted in the floppy disk drive. Note that the C> in the above commands is the MS-DOS system prompt and signifies that the default drive is the hard disk drive designated by C. The model is now ready for execution as described in Section 4 of this manual. A complete example of a simulation run is given in Section 5.

4. Executing the GPCSS Model

Before executing the model, data collection must be accomplished as described in the thesis cited in Section 2. Several parameters must be altered by editing the SLAM network source code file, GPCSS.DAT. It may also be necessary to change parts of the FORTRAN subroutines in file GPCSS.FOR. The procedures involved in changing these files are described in detail in Section 2 of this manual.

4.1 Model Execution

Be sure the default drive setting is B if a dual floppy system is being used or C if a hard drive system is being used. Also, the PC SLAM II Master diskette must be in disk drive A. To execute the model, type in GPCSS. The model will

begin to run and will prompt the user to enter the name of the translated model as follows :

```
Enter the name of the translated model:
```

The user's response should be GPCSS.TRA. The screen will clear and the following menu will appear :

```
THE FOLLOWING ARE YOUR OPTIONS FOR THIS SIMULATION:

            1 - Build a NEW Configuration File
            2 - Run a Simulation with an EXISTING
                Configuration File
            3 - EXIT

Enter the number of the desired option -->
```

4.2 Option 1

If option 1 is selected, the user is prompted to enter a file name for a new configuration file. The file name must conform to the MS-DOS format described in the MS-DOS manual. It is suggested that the first eight characters of the file name be used to identify the commissary being simulated and that the extension for the file name be .CNF. For example, if a simulation of the Lackland AFB Commissary is to be run, the file name could be LACKLAND.CNF.

After a file name has been entered, the program will display the following :

```
The following prompts will enable you to
configure a Commissary Store. The store will
be simulated using your inputs.

Enter the number of Parking Spots -->400
Enter the number of Large Carts ---->450
Enter the number of Small Carts ---->150
Enter the number of baggers -------->20
```

```
Enter the number of Queue Lines ---->2
Number of waiting Customers -------->38
Enter Opening Time ----------------->0830
Enter Closing Time ----------------->1800
```

(The values after the pointers are examples of possible responses and are included for clarity).

All of the prompts are self-explanatory, however two clarifications are necessary. First, the number of baggers is the number of baggers on duty at opening time. The number of waiting customers refers to the number of customers at the front door when the store is opened.

After the general information is input, the user is asked to verify that the data is correct. The following is displayed on the screen :

```
The Following are your inputs ...

     # Parking Spots     :   400
     # Large Carts       :   450
     # Small Carts       :   150
     # Baggers           :    20
     # Queue Lines       :     2
     # Waiting Customers :    38
     Opening Time        :  0830
     Closing Time        :  1800

Are they correct ? [0-No, 1-Yes] -->
```

(Again, the values after the pointers are included for clarity). If any of the data is incorrect, the user should respond with 0. The program will then have the user enter the correct data. Once all data is verified as correct, the user enters a 1 and the program proceeds to the next phase of user input.

The next data to be entered is the Front End information. The following is displayed to the screen :

```
You will now be prompted to enter FRONT END
information, i.e. the number of checkstands and
which queue lines they service.

* Enter the number of checkstands Queue Line 1
  is serviced by ---------------------------->10

  Enter the number of checkstands Queue Line 2
  is serviced by ---------------------------->10

  Enter the number of Queue Lines that will be
  open at start of business ------------------>2


* How many checkstands will be open for Queue Line 1
  at the start of business ------------------------>2

  How many checkstands will be open for Queue Line 2
  at the start of business ------------------------>2
```

*Note : This prompt appears twice since in this example a value of 2 was entered for the number of Queue Lines. This prompt will always appear N times where N is the number of Queue lines in the store being simulated.

After the Front End information has been entered, the user is asked to verify that the data is correct. The following appears on the screen :

```
The following are your inputs ...

Max # checkstands servicing Queue Line 1 : 10
Max # checkstands servicing Queue Line 2 : 10
# Queue Lines open at start of business  :  2
# Checkstands open for Queue Line 1 at start of business :  2
# Checkstands open for Queue Line 2 at start of business :  2

Are they correct ? [0-No, 1-Yes] -->
```

If any of the data is incorrect, 0 should be entered. The program will prompt the user to enter the correct data. Once

all data is verified as correct, the user enters a 1 and the program proceeds to the next and last phase of user input.

The last phase of user input is customer arrival rates to the store per minute. The following is displayed to the screen :

```
You will now be prompted to enter
Customer arrival rates ...

0830 Hrs to 0930 Hrs

Enter customer arrival rate ------->3.1

          .
          .
          .

1730 Hrs to 1800 Hrs

Enter customer arrival rate ------->2.7
```

After the data is entered, the following is displayed to the screen :

```
The following are your inputs ...

     FROM  TO     ARRIVAL RATE
     ----  --     ------------
     0830 0930        3.1
            .          .
            .          .
            .          .
     1730 1800        2.7

Are they correct ? [0-No, 1-Yes] -->
```

If any of the data is incorrect, 0 should be entered. The program will prompt the user to enter the correct data.

After all data has been verified as correct, the screen clears and the following is displayed :

```
Enter File Name for Output Statistics -->
```

The user must enter a file name that conforms to MS-DOS standards. This file will contain all output statistics from the simulation runs. It is recommended that the file name entered for the output statistics be the same as the file name of the configuration file except for the extension. For example, if the configuration file name is LACKLAND.CNF, the file name for the output statistics should be LACKLAND.OUT. Next, the following prompt appears on the screen :

```
Enter number of Runs desired -->
```

The user must enter the number of runs desired, keeping in mind that the model will only do up to 10 runs. The model begins execution at this point, and the following appears on the screen :

```
Run 1 of N
Simulation Begins ...
```

Where N is the number of runs the user requested.

Every 15 minutes of *simulated* time, the following message appears :

```
Hour: 1  No. in Store: 50  No. Cashiers: 4  Avg. Wait : .00
```

Hour specifies which hour of operation the model is in. No. in Store is the number of customers in the system. No. Cashiers is the number of cashiers on duty. Avg. Wait is the average waiting time of the customers in the checkout lines who were serviced in the last 15 minutes.

4.3 Option 2

If option 2 is selected, the user is prompted to enter the file name of the existing configuration file. For example, if the simulation for the Lackland AFB Commissary had been previously run and the user wanted to run it again, the user would enter the file name LACKLAND.CNF.

When option 2 is selected, the program does not ask the user to input any data. Instead, the data contained in the existing configuration file for each of the three phases is displayed on the screen, and the user is given the opportunity to make changes if so desired. The following will appear on the screen :

```
The Following will be used as inputs ...

        # Parking Spots      :   400
        # Large Carts        :   450
        # Small Carts        :   150
        # Baggers            :    20
        # Queue Lines        :     2
        # Waiting Customers  :    38
        Opening Time         :  0830
        Closing Time         :  1800

Are they correct ? [0-No, 1-Yes] -->
```

If any of the data is incorrect, the user should respond with 0. The program will then have the user enter the correct data. Once all data is verified as correct, the user enters a 1 and the program proceeds to verify the second phase of data contained in the existing configuration file.

To verify the Front End data and to give the user the chance to make changes to the existing Front end data, the following appears on the screen :

```
The following will be used as inputs ...

Max # checkstands servicing Queue Line 1 : 10
Max # checkstands servicing Queue Line 2 : 10
# Queue Lines open at start of business  :  2
# Checkstands open for Queue Line 1 at start of business :  2
# Checkstands open for Queue Line 2 at start of business :  2

Are they correct ? [0-No, 1-Yes] -->
```

If any of the data is incorrect, 0 should be entered. The program will prompt the user to enter the correct data. Once all data is verified as correct, the user enters a 1 and the program proceeds to verify the last data contained in the existing configuration file.

The customer arrival rate data is the last data in the existing configuration file that must be verified. The following is displayed to the screen :

```
     The following will be used as inputs ...

          FROM  TO     ARRIVAL RATE
          ----  --     ------------
          0830 0930        3.1
            .               .
            .               .
            .               .
          1730 1800        2.7

     Are they correct ? [0-No, 1-Yes] -->
```

If any of the data is incorrect, 0 should be entered. The program will prompt the user to enter the correct data.

As when option 1 was is chosen, the following prompts and messages appear :

Enter File Name for Output Statistics -->

The user must enter a file name that conforms to MS-DOS standards. This file will contain all output statistics from the simulation runs. It is recommended that the file name entered for the output statistics be the same as the file name of the configuration file except for the extension. For example, if the configuration file name is LACKLAND.CNF, the file name for the output statistics should be LACKLAND.OUT. Next, the following prompt appears on the screen :

Enter number of Runs desired -->

The user must enter the number of runs desired, keeping in mind that the model will only do up to 10 runs. The model begins execution at this point, and the following appears on the screen :

Run 1 of N

Simulation Begins ...

Where N is the number of runs the user requested.

Every 15 minutes of *simulated* time, the following message appears :

Hour: 1 No. in Store: 50 No. Cashiers: 4 Avg. Wait : .00

Hour specifies which hour of operation the model is in. No.

in Store is the number of customers in the system. No. Cashiers is the number of cashiers on duty. Avg. Wait is the average waiting time of the customers in the checkout lines who were serviced in the last 15 minutes.

4.4 Option 3

If option 3 is chosen, the following prompt will appear on the screen : Stop. Program Terminated.

4.5 Execution Time

The simulation will run for several minutes depending on the opening and closing times the user entered in the first phase of input since the model calculates the number of simulated minutes the store will be open from these times. For example, if the opening time is 0830 and the closing time is 1800, the model runs for 570 simulated minutes. When the model detects that closing time has occurred, the following message is shown on the screen :

Closing Front Door ...

No more customers will be created. However, the customers in the system when the closing time was detected will continue to flow through the store. When all of these customers have been serviced, the current simulation terminates. If more runs were requested by the user, the following will be displayed to the screen:

Run 2 of N

Simulation Begins ...

Where N is the number of runs the user requested.

As for the first run, every 15 minutes of *simulated* time, the following message appears :

Hour: 1 No. in Store: 50 No. Cashiers: 4 Avg. Wait : .00

After all N runs requested by the user have completed, the following prompts appear :

Enter File Name for Wait Time Plot -->

It is recommended that the names for this file be entered with a .PLT extension. For example, LACKLAND.PLT. This data file contains the mean estimates and confidence interval for the waiting times of customers as described in Chapter III. The user can use a plotting program such as is contained in SAS to plot these values.

After a file name for the waiting data has been entered, the following appears :

Enter File Name for # Cashiers Plot -->

As before, a file name that identifies the data with this simulation run should be entered. For example, LACKLAND.CSH would signify the data file contains cashier data for the Lackland Commissary simulation. As before, a program such as SAS could be used to plot the data if desired.

4.6 Obtaining Output Reports

Since the GPCSS model was designed to simulate many different configurations of commissaries, the Summary Reports produced by the PC SLAM Output Processor will contain some information which will be of no value to the user. For this reason, a tailored output report is produced by the model and is written to the output statistics file specified by the user as described in Sections 4.1 and 4.2 of this manual. The user can view this output report by invoking the MS-DOS command TYPE. If a printed copy of the report is desired, the PRINT command should be entered.

Additionally, the user can use a plotting program such as SAS to plot the waiting time data and number of cashiers data in the plot files described in sections 4.1 and 4.2

5. Example Run of the GPCSS Model

The following is an example of a complete run of the GPCSS Model. The Z-248 Computer system used was configured with a hard disk drive, therefore the default drive will be designated as C. All user input is capitalized and underlined. Comments about what is going on in the model execution are enclosed in square brackets [ ].

Prior to model execution, the PC SLAM II master diskette was inserted into disk drive A.

[To begin execution, the following command is entered :]

C>GPCSS

[The screen will clear and then the copyright message for SLAM II appears. The user is prompted to enter the filename of the translated model as follows :]

```
Enter file name of translated model:GPCSS.TRA
```

[The screen clears again and the main menu will appear.]

```
THE FOLLOWING ARE YOUR OPTIONS FOR THIS SIMULATION:

  1 - Build a NEW Configuration File
  2 - Run a simulation using an EXISTING
      Configuration File
  3 - EXIT

Enter the number of the desired option -->1
```

[Since 1 was entered, the user is prompted to enter a file name.]

```
Enter the Name of the NEW File -->WPAT.CNF
```

[The following prompts appear sequentially :]

```
Enter the number of Parking Spots -->450
Enter the number of Large Carts ---->450
Enter the number of Small Carts ---->150
Enter the number of Baggers -------->20
Enter the number of Queue Lines ---->2
Number of waiting Customers -------->38
Enter Opening Time ----------------->0830
Enter Closing Time ----------------->1800
```

[Now the user is asked to verify the data just entered.]

```
The following are your inputs ...

# Parking Spots     :        450
# Large Carts       :        450
# Small Carts       :        150
# Baggers           :         20
```

```
# Queue Lines        :          2
Opening Time         :        830
Closing Time         :       1800

Are they correct ? [0-No, 1-Yes] --> 0
```

[Since 0 was entered, the user wishes to correct an entry. The following now appears :]

```
  Which of the following is incorrect ? :

      1 - Number of Parking Spots
      2 - Number of Large Carts
      3 - Number of Small Carts
      4 - Number of Baggers
      5 - Number of Queue Lines
      6 - Number of Waiting Customers
      7 - Opening Time
      8 - Closing Time

Enter the number of the incorrect value --> 6
Re-enter the number of Waiting Customers --> 40


After correction, your inputs are now ...

# Parking Spots      :        450
# Large Carts        :        450
# Small Carts        :        150
# Baggers            :         20
# Queue Lines        :          2
Opening Time         :        830
Closing Time         :       1800

Are they correct ? [0-No, 1-Yes] --> 1
```

[At this point, the GENERAL information has been input and verified. The next phase of input is FRONT END information.]

```
You will now be prompted to enter FRONT END
information, i.e. number of checkstands and
which queue lines they service.

Enter the number of checkstands Queue Line 1
is serviced by ---------------------> 10
```

```
     Enter the number of checkstands Queue Line 2
     is serviced by ----------------------> 10

     Enter the number of Queue Lines that will be
     open at start of business -------------> 2

     How many checkstands will be open for Queue Line 1
     at the start of business -----------------> 2

     How many checkstands will be open for Queue Line 2
     at the start of business -----------------> 2
```

[As before, the user is asked to verify the data just input.]

```
The following are your inputs ...

Max # checkstands servicing Queue Line  1 : 10
Max # checkstands servicing Queue Line  2 : 10
# Queue Lines open at start of business   :  2
# checkstands open for Queue Line 1 at start of business : 2
# checkstands open for Queue Line 2 at start of business : 2

Are they correct ? [0-No, 1-Yes] --> 3


Invalid response ...
Hit RETURN to continue -->
```

[The user accidently hit 3 instead of 0 or 1, so the program notified the user they made an invalid response. Control returns to the same point of user input and continues.]

```
The following are your inputs ...

Max # checkstands servicing Queue Line  1 : 10
Max # checkstands servicing Queue Line  2 : 10
# Queue Lines open at start of business   :  2
# checkstands open for Queue Line 1 at start of business : 2
# checkstands open for Queue Line 2 at start of business : 2

Are they correct ? [0-No, 1-Yes] --> 1
```

[After FRONT END information is input and verified, the last phase of input, arrival data, begins.]

```
     You will now be prompted to enter
     Customer arrival rates per minute ...
```

0830 Hrs to 0930 Hrs :

Enter customer arrival rate -------> 3.55

0930 Hrs to 1030 Hrs :

Enter customer arrival rate -------> 4.93

1030 Hrs to 1130 Hrs :

Enter customer arrival rate -------> 5.23

1130 Hrs to 1230 Hrs :

Enter customer arrival rate -------> 5.25

1230 Hrs to 1330 Hrs :

Enter customer arrival rate -------> 4.46

1330 Hrs to 1430 Hrs :

Enter customer arrival rate -------> 4.18

1430 Hrs to 1530 Hrs :

Enter customer arrival rate -------> 4.26

1530 Hrs to 1630 Hrs :

Enter customer arrival rate -------> 4.7

1630 Hrs to 1730 Hrs :

Enter customer arrival rate -------> 3.67

1730 Hrs to 1800 Hrs :

Enter customer arrival rate -------> 3.60

The following are your inputs ...

FROM	TO	ARRIVAL RATE
830	930	3.55
930	1030	4.93
1030	1130	5.23
1130	1230	5.25
1230	1330	4.46
1330	1430	4.18
1430	1530	4.26

```
1530  1630      4.70
1630  1730      3.67
1730  1630      3.60

Are they correct ? [0-No, 1-Yes] --> 1
```

[All data has been input and verified. The user is now prompted to enter a file name for output statistics and to specify how many runs are desired.]

```
Enter File Name for Output Statistics --> LACKLAND.CNF
Enter number of Runs desired --> 5
```

[The simulation runs begin now and the following is displayed to the screen.]

```
Run 1 of 1

Simulation Begins ...

Hour: 1  No. in Store: 79  No. Cashiers : 4 Avg.Wait: .00
      .
      .
      .
Hour:10  No. in Store:244  No. Cashiers :18 Avg.Wait:9.29
```

[When closing time is reached in the simulation, the following message appears :]

```
Closing Front Door ...
```

[The above process is continued until all five simulations have completed. Then the user is prompted to enter the name to be assigned to the plotting data files.]

```
Enter File Name for Wait Time Plot --> LACKLAND.PLT
Enter File Name for # Cashiers Plot --> LACKLAND.CSH
```

[The user can now obtain output reports as discussed in Section 4.6 of this manual.]

Bibliography

1. Ben-Akiva, Moshe and others. "Dynamic Model of Peak Period Traffic Congestion, with Elastic Arrival Rates," Transportation Science, 20 : 164-181 (August 1986).

2. Cinlar, Erhan. Introduction to Stochastic Processes. Englewood Cliffs NJ: Prentice-Hall, 1975.

3. De Palma, A. and others. "Stochastic Equilibrium Model of Peak Period Traffic Congestion," Transportation Science, 17 : 430-453 (June 1983).

4. Dorough, Capt Robert E. and Capt Robert E. Holliway. A Simulation Study of Checkout Operations at the Wright-Patterson AFB Commissary. MS Thesis, AFIT/LSSR/6-80. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, June 1980 (AD-A087).

5. Gordon, Geoffrey. System Simulation. Englewood Cliffs NJ: Prentice-HAll, 1978.

6. Gross, Donald and Carl M. Harris. Fundamentals of Queuing Theory. New York: John Wiley and Sons, 1974.

7. Hillier, Frederick S. and Gerald J. Lieberman. Introduction to Operations Research. Oakland CA: Holden-Day, 1986.

8. HQ AFCOMS. Thesis topics briefing. Kelly AFB, TX, 23 March 1987.

9. Duell, Gary C., Chief, Merchandising and Marketing Branch. Personal interview. HQ AFCOMS, Kelly AFB TX, 31 August 1987.

10. Johnson, Don, Store Manager, Wright-Patterson AFB Commissary. Personal Interviews. Wright-Patterson AFB, OH, 2 October 1987 through 25 Nov 1987.

11. Jones, Kenneth, Manpower and Organization Review(XPM). Telephone interview. HQ AFCOMS, Kelly AFB TX, 30 September 1987.

12. Jones, Michael T. "Quickening the Queue in Grocery Stores," Interfaces, 10 : 90-92 (June 1980).

13. Krajewski, L. J. and others. "Shift Scheduling in Banking Operations : A Case Application," Interfaces, 10 : 1-8 (April 1980).

14. Litko, Major Joseph R. Class handout distributed in OPER 666, Military Systems Simulation. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, February 1987.

15. Meyer, N. Dean. " The Role of Management Science in Office Automation," Interfaces, 10 : 72-76 (February 1980).

16. Mendenhall, William and others. Mathematical Statistics with Applications. Duxbury Press: Boston, 1986.

17. Moore, Albert. Class lectures and handouts in MATH686, Nonparametric Statisics. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, November 1987.

18. Neter, John and others. Applied Linear Statistical Models. Homewood IL: Irwin, 1985.

19. Paul, R. J. and R. E. Stevens. "Staffing Service Activities with Waiting Line Models," Decision Sciences, 2 :206-218 (April 1980).

20. Polk, Lt Col Stan, Chief Cost Division. Personal interviews. HQ AFCOMS, Kelly AFB TX. 9 April through 1 September 1987.

21. Pritsker, A. Alan B. and C. Elliot Sigal. Management Decision Making, A Network Simulation Approach. Englewood Cliffs NJ: Prentice-Hall, 1983.

22. Pritsker, A. Alan B. Introduction to Simulation and SLAM II. West Lafayette IN: Systems Publishing, 1986.

23. SAS Institute Inc. SAS User's Guide: Statistics, Version 5 Edition. Cary, NC:SAS Institute Inc.

24. Solheim, Major Mark H., Commissary Officer, Wright-Patterson AFB Commissary. Personal Interviews. Wright-Patterson AFB, OH, 2 October 1987 through 25 November 1987.

25. Solomon, Susan L. "Building Modelers: Teaching the Art of Simulation," Interfaces, 10 : 65-72 (April 1980).

26. "USAF Almanac 1986," Air Force Magazine ,69 : 115-116 (May 1986).

27. Wynne, Bye. "Behavioral Science--Keys to Successful Management Science Modeling," Interfaces, 9 : 69-74 (August 1979).

Vita

Captain Roger D. Moulder was born on 7 May 1951 in Smiths Grove, Kentucky. He graduated from high school in Bowling Green, Kentucky in 1969 and enlisted in the Air Force in October 1970. In 1981 he was selected for the Air Force's Airmen's Education and Commissioning Program through which he attended the University of Kentucky. In May 1983 he received the degree of Bachelor of Science in Computer Science and Mathematics. After graduation, he attended the Air Force's Officer Training School and received his commission in the USAF on 17 August 1983. He was then assigned to the Air Force Wright Aeronautical Laboratories (AFWAL), Wright-Patterson AFB, OH, where he worked in the Avionics Laboratory as a computer research scientist until he entered the School of Engineering, Air Force Institute of Technology, in June of 1986.

Permanent address: Lot 68 Cabana Estates
Bowling Green, Kentucky 42101

LMED

8